PHOTOGRAPH THIS SHEET

AD-A280 169

DTIC ACCESSION NUMBER

LEVEL

INVENTORY

Image Understanding Workshop
DOCUMENT IDENTIFICATION
Jan 92

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

DISTRIBUTION STATEMENT

ACCESSION FOR
NTIS GRA&I
DTIC TRAC
UNANNOUNCED
JUSTIFICATION

BY
DISTRIBUTION/
AVAILABILITY CODES

DISTRIBUTION	AVAILABILITY AND/OR SPECIAL	
A-1		

DISTRIBUTION STAMP

DTIC
ELECTE
JUN 7 1994
S C D

DATE ACCESSIONED

DATE RETURNED

9416121

DATE RECEIVED IN DTIC

REGISTERED OR CERTIFIED NUMBER

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-FDAC

DTIC FORM 70A APR 87 DOCUMENT PROCESSING SHEET

PREVIOUS EDITIONS MAY BE USED UNTIL STOCKS EXHAUSTED

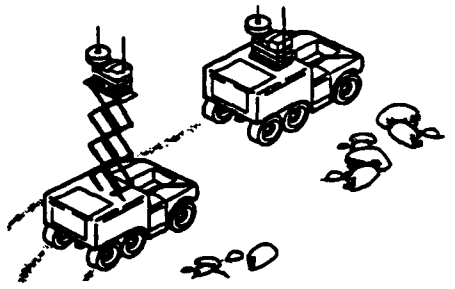
94 5 27 087

Proceedings:

IMAGE UNDERSTANDING WORKSHOP

Image Understanding Applications

UNMANNED GROUND VEHICLE (UGV)



RADIUS

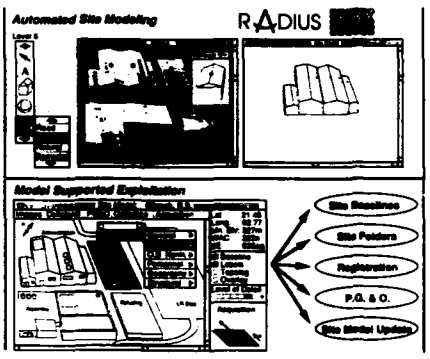
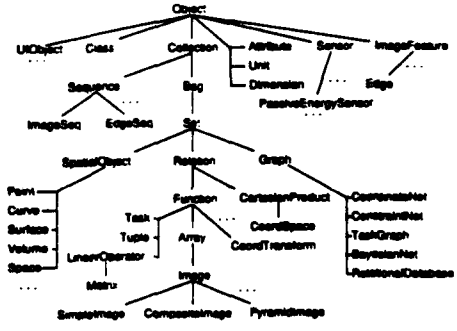


IMAGE UNDERSTANDING ENVIRONMENT



Sponsored by:

Defense Advanced Research Projects Agency
Software and Intelligent Systems Technology Office

January 1992

Image Understanding Workshop

Proceedings of a Workshop
held at
San Diego, California

January 26 - 29, 1992

Sponsored by:

Defense Advanced Research Projects Agency
Software and Intelligent Systems Technology Office

This document contains copies of reports prepared for the DARPA Image Understanding Workshop. Included are Principal Investigator Reports and technical results from both the basic and strategic computing programs within DARPA/SISTO sponsored projects and certain technical reports from selected scientists from other organizations.

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

Distributed by
Morgan Kaufmann Publishers, Inc.
2929 Campus Drive, Suite 260
San Mateo, CA 94403
ISBN 1-55860-244-5
Printed in the United States of America

TABLE OF CONTENTS

AUTHOR INDEX ix

ACKNOWLEDGEMENTS xi

FOREWORD xiii

SECTION I : PRINCIPAL INVESTIGATORS REPORTS

"USC Image Understanding Research: 1990-1991," R. Nevatia, K. Price, and G. Medioni, University of Southern California 3

"Image Understanding Research at SRI International," Martin A. Fischler and Robert C. Bolles, SRI International 27

"Image Understanding: Intelligent Decisions," Thomas O. Binford, Stanford University 37

"CMU Image Understanding Program," Takeo Kanade, Carnegie Mellon University 51

"Progress in Computer Vision at the University of Massachusetts," Edward M. Riseman and Allen R. Hanson, University of Massachusetts 63

"Progress in Image Understanding at MIT," W. E. L. Grimson, B. K. P. Horn, and T. Poggio and staff, Massachusetts Institute of Technology 69

"Image Understanding Research at Columbia University," Peter K. Allen, John R. Kender, Terrance E. Boult, and Shree K. Nayar, Columbia University 83

"Maryland Progress in Image Understanding," Yiannis Aloimonos, Rama Chellappa, Larry S. Davis, and Azriel Rosenfeld, University of Maryland 97

"Image Understanding Research at Rochester," Dana Ballard, Christopher Brown, and Randal Nelson, University of Rochester 109

"Image Understanding Research at GE," J. L. Mundy, GE Corporate Research and Development 117

"IU at UI: An Overview of Research During 1990-91," Narendra Ahuja and Thomas Huang, University of Illinois 127

"An Active Observer," Ruzena Bajcsy, University of Pennsylvania 137

"Vision-Based Navigation," William B. Thompson, University of Utah; and Herbert L. Pick, Jr., University of Minnesota 149

"Research at Brown University," David B. Cooper, Thomas L. Dean, and William A. Wolovich, Brown University ... 153

SECTION II: CURRENT IU APPLICATIONS

"The OSD Tactical Unmanned Ground Vehicle Program," LTC Erik G. Mettala, PhD, Defense Advanced Research Projects Agency 159

"RADIUS: The Government Viewpoint," Donald J. Gerson, Office of Research and Development 173

"RADIUS: Research and Development for Image Understanding Systems Phase 1," Jeffrey Edwards, Shirley Gee, Arthur Newman, Randall Onishi, Alan Parks, Mark Sleeth, and Felicia Vilnrotter, Hughes Aircraft Company 177

"The Image Understanding Environments Program," J. Mundy, GE Corporate Research and Development; T. Binford, Stanford University; T. Boult, Columbia University; A. Hanson and R. Beveridge, University of Massachusetts; R. Haralick and V. Ramesh, University of Washington; C. Kohl, Amerinex Artificial Intelligence, Inc. ; D. Lawton, Georgia Institute of Technology; D. Morgan, Advanced Decision Systems; K. Price, University of Southern California; and T. Strat, SRI International 185

"The RADIUS Common Development Environment," J. L. Mundy and R. Welty, GE Corporate Research and Development; L. Quam and T. Strat, SRI International; and W. Bremner, M. Horwedel, D. Hackett, and A. Hoogs, GE Military and Data Systems Operation 215

SECTION III: FUTURE IU APPLICATIONS

"Understanding Synthetic Aperture Radar Images," Rama Chellappa, University of Maryland; Ed Zelnio, Wright-Patterson AFB; and Eric Rignot, California Institute of Technology 229

"Image Understanding Research for Automatic Target Recognition," Bir Bhanu, University of California; and Terry L. Jones, Night Vision and Electro-Optics Directorate 249

"ATR: Problems and Possibilities for the IU Community," Jan D. Wald and David B. Krig, Alliant Techsystems, Inc. ; and Trent DePersia, Defense Advanced Research Projects Agency 255

"Building a Cartographic Infra-Structure: Simulation and Cartography," David M. McKeown, Jr. , Carnegie Mellon University; and George E. Lukes, U. S. Army Topographic Engineering Center 265

"Status and Current Research in the Image Understanding Architecture Effort," Charles Weems, Martin Herbordt, and Michael Scudder, University of Massachusetts; and James Burrill, Richard Lerner, and Thomas Williams, Amerinex Artificial Intelligence, Inc. 269

SECTION IV: TECHNICAL PAPERS

Subsection A: Physics-based and Low Level Vision

"Perceptual Organization Without Edges," J. Brian Subirana-Vilanova and Kah Kay Sung, Massachusetts Institute of Technology 289

"Contour Matching Using Local Affine Transformations," Ivan A. Bachelder and Shimon Ullman, Massachusetts Institute of Technology 299

"HyperBF Networks for Gender Classification," R. Brunelli, Istutato per la Ricerca Scientifica e Tecnologica; and T. Poggio, Massachusetts Institute of Technology 311

"Sensitivity of Feature Configuration in Viewpoint Determination," Karen T. Sutherland, University of Minnesota 315

"Strategies for Localization," Marian R. Heinrichs, Kip Smith, Herbert L. Pick, Jr. , and Bonnie H. Bennett, University of Minnesota; and William B. Thompson, University of Utah 321

"Feature Extraction for Localization," Steven L. Savitt, University of Minnesota; and Thomas C. Henderson and Thomas L. Colvin, University of Utah 327

"Colored Interreflections and Shape Recovery," Shree K. Nayar and Yitao Gong, Columbia University 333

"Consensus by Decomposition: A Paradigm for Fast High Breakdown Point Robust Estimation," Doron Mintz, LSI Logic Corporation; Peter Meer, Rutgers University; and Azriel Rosenfeld, University of Maryland 345

"Correcting Chromatic Aberrations Using Image Warping," Terrance E. Boult, Columbia University; and George Wolberg, City University of New York 363

"On Finding the Ends of SHGCs in an Edge Image," Hiroaki Sato, Canon, Inc. Information Systems Research Center; and Thomas O. Binford, Stanford University 379

Subsection B: Stereo

"Stereo Using the Displacement Representation," Steve White, Massachusetts Institute of Technology 391

"Recovering LSHGCs and SHGCs from Stereo," Ronald C. -K. Chung and Ramakant Nevatia, University of Southern California 401

"A Multiple-baseline Stereo Method," Takeo Kanade, Masatoshi Okutomi, , and Tomoharu Nakahara, Carnegie Mellon University 409

"Hierarchical Stereo Matching using Feature Groupings," V. Venkateswar, Texas Instruments; and R. Chellappa, University of Maryland 427

"On Diffuse Reflection and Photometric Stereo," Lawrence B. Wolff, The Johns Hopkins University 437

Subsection C: Motion

"Motion Tracking on the Spatiotemporal Surface," H. H. Baker and T. D. Garvey, SRI International 451

"The Factorization Method for the Recovery of Shape and Motion from Image Streams," Carlo Tomasi, Cornell University; and Takeo Kanade, Carnegie Mellon University 459

"Integrated 3D Recovery and Visualization of Flight Image Sequences," Sanghoon Sull and Narendra Ahuja, University of Illinois 473

"Refinement of Noisy Correspondence Using Feedback from 3-D Motion," Yong Cheol Kim and Keith Price, University of Southern California 479

"Structure from Chronogeneous Motion: A Summary," Wolfgang Otto Franzen, University of Southern California 487

"Motion Segmentation Using Singular Value Decomposition," Terrance E. Boult and Lisa Gottesfeld Brown, Columbia University 495

"Recursive Multi-Frame Structure from Motion Incorporating Motion Error," J. Inigo Thomas and J. Oliensis, University of Massachusetts 507

"A Computational Model for Binocular Motion Processing," Poornima Balasubramanyam and M. A. Snyder, University of Massachusetts 515

"Behavioral Visual Motion Analysis," Yiannis Aloimonos, Zoran Duric, Cornelia Fermuller, Liuqing Huang, Ehud Rivlin, and Rajeev Sharma, University of Maryland 521

"Motion and Structure Factorization and Segmentation of Long Multiple Motion Image Sequences," Chris Debrunner and Narendra Ahuja, University of Illinois 543

"Motion Analysis of Articulated Objects with Applications to Human Ambulatory Patterns," Richard J. Qian and Thomas S. Huang, University of Illinois 549

"Temporal Texture Analysis," Ramprasad Polana and Randal C. Nelson, University of Rochester 555

Subsection D: Shape Recovery and Analysis

"Direct Method for Reconstructing Shape from Shading," Paul Dupuis, Brown University; and John Oliensis, University of Massachusetts 563

"Extracting Geometric Shapes From a Set of Points," Alicia Abella, Columbia University 573

"On Seeing Spaghetti: A Novel Self-Adjusting Seven Parameter Hough Space for Analyzing Flexible Extruded Objects," John R. Kender, Columbia University; and Rick Kjeldsen, IBM Thomas J. Watson Research Center 585

"Shape from Focus System for Rough Surfaces," Shree K. Nayar, Columbia University 593

"Hierarchical Decomposition and Axial Representation of Shape," Hillel Rom and Gerard Medioni, University of Southern California 607

"Reconstructing Surfaces from Unstructured 3D Points," P. Fua, SRI International; and P. Sander, INRIA Sophia-Antipolis 615

"Recovery of 3-D Objects with Multiple Curved Surfaces from 2-D Contours," Fatih Ulupinar and Ramakant Nevatia, University of Southern California 627

"Recovering 3-D Wire Frames From Line Drawings," Martin A. Fischler and Yvan G. Leclerc, SRI International 635

"Can Too Much Perspective Spoil the View? A Case Study in 2D Affine Versus 3D Perspective Model Matching," J. Ross Beveridge and E. M. Riseman, University of Massachusetts 655

Subsection E: Object Recognition and Scene Analysis

"Recognition of 3-D Objects from 2-D Groupings," Fridtjof Stein and Gerard Medioni, University of Southern California 667

"Matching Complex Images to Multiple 3D Objects using View Description Networks," J. Brian Burns and Edward M. Riseman, University of Massachusetts 675

"Noise Resistant Projective and Affine Invariants," Isaac Weiss, University of Maryland 683

"Polynomial-Time Object Recognition in the Presence of Clutter, Occlusion, and Uncertainty," Todd A. Cass, Massachusetts Institute of Technology 693

"Data and Model-Driven Selection using Color Regions," Syeda Tanveer Fathima, Massachusetts Institute of Technology 705

"Space Efficient 3D Model Indexing," David W. Jacobs, Massachusetts Institute of Technology 717

"3-D Model Alignment Without Computing Pose," Joseph L. Mundy and Richard P. Welty, GE Corporate Research and Development; Michael H. Brill, Science Applications International Corporation; and Paul M. Payton and Eamon B. Barrett, Lockheed Missiles and Space Company 727

"Application of Pose Determination Techniques to Model Extension and Refinement," Rakesh Kumar and Allen R. Hanson, University of Massachusetts 737

"Posterior Marginal Pose Estimation," William M. Wells III, Massachusetts Institute of Technology 745

"Model-Based Object Pose in 25 Lines of Code," Daniel F. DeMenthon and Larry S. Davis,

University of Maryland 753

"Representation Issues in Bayesian Decision Theory for Planning and Active Perception," Thomas Dean and Jak Kirman, Brown University 763

"Bounded and Unbounded Implicit Polynomial Curves and Surfaces, Mahalanobis Distances, and Geometric Invariants, for Robust Object Recognition," Daniel Keren, Jayashree Subrahmonia, and David B. Cooper, Brown University; and Gabriel Taubin, IBM T. J. Watson Research Center 769

"BUILDER-I: a System for the Extraction of SHGC Objects in an Edge Image," Hiroaki Sato, Canon, Inc. Information Systems Research Center; and Thomas O. Binford, Stanford University 779

"An Example of 3-D Interpretation of Images using Bayesian Networks," Wally Bishop Mann and Thomas O. Binford, Stanford University 793

Subsection F: Aerial Photo Matching and Interpretation

"Utilization of Multispectral Imagery for Cartographic Feature Extraction," Stephen J. Ford and David M. McKeown, Jr., Carnegie Mellon University 805

"The Direct Computation of Height from Shading," Yvan G. Leclerc and Aaron F. Bobick, SRI International 821

"The Interpretation of Synthetic Aperture Radar Images Using Projective Invariants and Deformable Templates," Aaron J. Heller, David M. LaRocque, and Joseph L. Mundy, GE CorporateResearch and Development 831

"Refinement of Disparity Estimates Through the Fusion of Monocular Image Segmentations," David M. McKeown, Jr. and Frederic P. Perlant, Carnegie Mellon University 839

"Tools for Acquiring Spatial and Functional Knowledge in Aerial Image Analysis," Wilson A. Harvey, Matthew Diamond, and David M. McKeown, Jr., Carnegie Mellon University 857

"On the Uniqueness of Correspondence under Orthographic and Perspective Projections," Ronen Basri, Massachusetts Institute of Technology 875

"Iterative TIN Generation from Digital Elevation Models," Michael F. Polis and David M. McKeown, Jr., Carnegie Mellon University 885

"A Novel Image Registration Algorithm," Qinfen Zheng, University of Southern California; and Rama Chellappa, University of Maryland 899

"Calibration of Cameras using the Essential Matrix," Richard Hartley, GE Corporate Research and Development 911

"Single Plane Model Extension using Projective Transformations," Robert T. Collins, University of Massachusetts 917

Subsection G: Active Vision and Visual Strategies

"Where to Look Next using a Bayes Net: An Overview," Raymond D. Rimey, University of Rochester 927

"Learning Knowledge-Directed Visual Strategies," Bruce A. Draper, Allen R. Hansen, and Edward M. Riseman, University of Massachusetts 933

"Sensor Planning in an Active Robotic Work Cell," Steven Abrams and Peter K. Allen, Columbia University 941

Subsection H: Robotics Applications

"Planning for Perception in Robot Driving," Douglas A. Reece and Steven A. Shafer, Carnegie Mellon University 953

"Combining Artificial Neural Networks and Symbolic Processing for Autonomous Robot Guidance," Dean A. Pomerleau, Jay Gowdy, and Charles E. Thorpe, Carnegie Mellon University 961

"Vehicle-Type Motion Modeling and Analysis," Yuncai Liu and Thomas S. Huang, University of Illinois 969

"Proximity Detection Using a Filter Tuned in Three-Space," Ian Horswill, Massachusetts Institute of Technology 973

"Stealth Terrain Navigation With Bounding Overwatch," Y. Ansel Teng, Daniel DeMenthon, and Larry S. Davis, University of Maryland..... 979

"Three Dimensional Object Tracking and Recognition Using a Dual-Drive Hybrid Controller," Kelly A. Korzeniowski and William A. Wolovich, Brown University 991

"Towards an Assembly Plan from Observation: Task recognition with polyhedral objects," Katsushi Ikeuchi and Takashi Suehiro, Carnegie Mellon University 997

"Tracking, Detection and 3D Representation of Potential Obstacles Using Affine Constraints," Harpreet S. Sawhney and Allen R. Hanson, University of Massachusetts 1009

"Trajectory Filtering and Prediction for Automated Tracking and Grasping of a Moving Object," Peter K. Allen, Aleksandar Timcenko, Billibon Yoshimi, and Paul Michelman, Columbia University *1019*

Subsection I: IU Software and Parallel Processing

"Overcoming the Barriers to Architecture-Independent Image Processing," Jon A. Webb, Carnegie Mellon University.....1037

"Parallel Linear Feature Extraction," Craig Reinhart and Ramakant Nevatia, University of Southern California 1049

"Parallel Algorithms for Stereo and Image Matching," Ashfaq Khokhar and Viktor K. Prasanna, University of Southern California 1057

AUTHOR INDEX

Abella, Alicia 573
Abrams, Steven 941
Ahuja, Narendra 127, 473, 543
Allen, Peter K. 83, 941, 1019
Aloimonos, Yiannis 97, 521

Bachelder, Ivan A. 299
Bajcsy, Ruzena 137
Baker, H. H. 451
Balasubramanyam, Poornima 515
Ballard, Dana 109
Barrett, Eamon B. 727
Basri, Ronen 875
Bennett, Bonnie H. 321
Beveridge, J. Ross 185, 655
Bhanu, Bir 249
Binford, Thomas O. 37, 185, 379, 779, 793
Bobick, Aaron F. 821
Bolles, Robert C. 27
Boult, Terrance E. 83, 185, 363, 495
Bremner, W. 215
Brill, Michael H. 727
Brown, Christopher 109
Brown, Lisa Gottesfeld 495
Brunelli, R. 311
Burns, J. Brian 675
Burrill, James 269

Cass, Todd A. 693
Chellappa, Rama 97, 229, 427, 899
Chung, Ronald C.-K. 401
Collins, Robert T. 917
Colvin, Thomas L. 327
Cooper, David B. 153, 769

Davis, Larry S. 97, 753, 979
Dean, Thomas L. 153, 763
Debrunner, Chris 543
DeMenthon, Daniel F. 753, 979
DePersia, Trent 255
Diamond, Matthew 857
Draper, Bruce A. 933
Dupuis, Paul 563
Duric, Zoran 521

Edwards, Jeffrey 177

Fathima, Syeda Tanveer 705
Fermuller, Cornelia 521
Fischler, Martin A. 27, 635
Ford, Stephen J. 805
Franzen, Wolfgang Otto 487
Fua, P. 615

Garvey, T. D. 451
Gee, Shirley 177
Gerson, Donald J. 173
Gong, Yitao 333
Gowdy, Jay 961
Grimson, W. E. L. 69

Hackett, D. 215
Hanson, Allen R. 63, 185, 737, 933, 1009
Haralick, R. 185
Hartley, Richard 909
Harvey, Wilson A. 857
Heinrichs, Marian R. 321
Heller, Aaron J. 831
Henderson, Thomas C. 327
Herbordt, Martin 269
Hoogs, A. 215
Horn, B. K. P 69
Horswill, Ian 973
Horwedel, M. 215
Huang, Liuqing 521
Huang, Thomas S. 127, 549, 969

Ikeuchi, Katsushi 997

Jacobs, David W. 717
Jones, Terry L. 249

Kanade, Takeo 51, 409, 459
Kender, John R. 83, 585
Keren, Daniel 769
Khokhar, Ashfaq 1057
Kim, Yong Cheol 479
Kirman, Jak 763
Kjeldsen, Rick 585
Kohl, C. 185
Korzeniowski, Kelly A. 991
Krig, David B. 255
Kumar, Rakesh 737

LaRocque, David M. 831
Lawton, D. 185
Leclerc, Yvan G. 635, 821
Lerner, Richard 269
Liu, Yuncai 969
Lukes, George E. 265

Mann, Wally Bishop 793
McKeown, Jr., David M. 265, 805, 839, 857, 885
Medioni, Gerard 3, 607, 667
Meer, Peter 345
Mettala, LTC Erik G. 159
Michelman, Paul 1019

Mintz, Doron 345
Morgan, D. 185
Mundy, Joseph L. 117, 185, 215, 727, 831

Nakahara, Tomoharu 409
Nayar, Shree K. 83, 333, 593
Nelson, Randal C. 109, 555
Nevatia, Ramakant 3, 401, 627, 1049
Newman, Arthur 177

Okutomi, Masatoshi 409
Oliensis, John 507, 563
Onishi, Randall 177

Parks, Alan 177
Payton, Paul M. 727
Perlant, Frederic P. 839
Pick, Jr., Herbert L. 149, 321
Poggio, T. 69, 311
Polana, Ramprasad 555
Polis, Michael F. 885
Pomerleau, Dean A. 961
Prasanna, Viktor K. 1057
Price, Keith 3, 185, 479

Qian, Richard J. 549
Quam, L. 215

Ramesh, V. 185
Reece, Douglas A. 953
Reinhart, Craig 1049
Rignot, Eric 229
Rimey, Raymond D. 927
Riseman, Edward M. 63, 655, 675, 933
Rivlin, Ehud 521
Rom, Hillel 607
Rosenfeld, Azriel 97, 345

Sander, P. 615
Sato, Hiroaki 379, 779
Savitt, Steven L. 327
Sawhney, Harpreet, S. 1009
Scudder, Michael 269
Shafer, Steven A. 953
Sharma, Rajeev 521
Sleeth, Mark 177
Smith, Kip 321
Snyder, M. A. 515
Stein, Fridtjof 667
Strat, T. 185, 215
Subirana-Vilanova, J. Brian 289
Subrahmonia, Jayashree 769
Suehiro, Takashi 997
Sull, Sanghoon 473
Sung, Kah Kay 289
Sutherland, Karen T. 315

Taubin, Gabriel 769
Teng, Y. Ansel 979
Thomas J. Inigo 507
Thompson, William B. 149, 321
Thorpe, Charles E. 961
Timcenko, Aleksandar 1019
Tomasi, Carlo 459

Ullman, Shimon 299
Ulupinar, Fatih 627

Venkateswar, V. 427
Vilnrotter, Felicia 177

Wald, Jan D. 255
Webb, Jon A. 1037
Weems, Charles 269
Weiss, Isaac 683
Wells III, William M. 745
Welty, Richard P. 215, 727
White, Steve 391
Williams, Thomas 269
Wolberg, George 363
Wolff, Lawrence B. 437
Wolovich, William A. 153, 991

Yoshimi, Billibon 1019

Zelnio, Ed 229
Zheng, Qinfen 899

ACKNOWLEDGEMENTS

The twentyfirst DARPA Image Understanding Workshop was held in San Diego, California on January 27-29, 1992. The workshop was sponsored by the Software and Intelligent Systems Technology Office of the Defense Advanced Research Projects Agency (DARPA). Dr. Ram Nevatia, University of Southern California, was the conference chairman. He was responsible for the organization of the technical sessions, and for the special session on IU architectures.

The first section of these proceedings is comprised of expanded research reviews presented by the principal investigators of the research institutions included in the DARPA program. The second and third sections deal with current and future applications of IU. The fourth section contains technical papers prepared by members of the technical staffs of each institution. Although these technical papers were not presented due to lack of time, they are included to present a complete record of recent results in the computer vision field.

The pictures appearing on the cover were selected to illustrate the conference theme "Image Understanding Applications." The RADIUS picture was submitted by Randy Onishi of Hughes, the Unmanned Ground Vehicle sketch by LTC Erik G. Mettala of DARPA/SISTO, and the class hierarchy diagram for the Image Understanding Environment (IUE) by Doug Morgan of ADS.

The conference coordinator was Lois Hollan of PRC, Inc./Meridian, a one-person team who handled the conference arrangements, the coordination and design of the proceedings, and the thousand of small details that go into putting on a conference. Of great assistance in coordinating the proceedings was William R. O'Neill of Meridian Corporation

FOREWORD

Between the idea
And the reality...
Falls the Shadow

The Hollow Men, T.S. Elliot

To paraphrase Elliot, between the IU research and the applications, falls the shadow. In recognition of this, the theme for the 1992 DARPA Image Understanding Workshop is "Image Understanding Applications." We are back to the theme stressed by Bob Simpson in the 1985 DARPA Workshop Proceedings when he warned, "... We need to identify real defense applications that have been made possible by the basic IU research. Otherwise you will have difficulty justifying the continuation of IU." Indeed, there is even more need today to show how the IU research of the past decade or more can be made to pay off in applications.

In response to this need, we have scheduled a Current Applications session, with presentations on the Unmanned Ground Vehicle (UGV), RADIUS, the Image Understanding Environment (IUE), and the Cartographic Modeling Environment Enhanced (CMEE). The Future Applications session has presentations on SAR, Automatic Target Recognition, the Rapid Geographic Database, and emerging applications in Bomb Damage Assessment, Image Dissemination, and Flexible Template Generation.

The first robotic vehicle in the history of the United States will be fielded as a result of the Joint Unmanned Ground Vehicles program. This program has three major thrusts: 1) building and early user testing of a Surrogate Teleoperated Vehicle; 2) engineering manufacturing and development of a Tactical Unmanned Ground Vehicle; and 3) Unmanned Ground Vehicle technology enhancement and exploitation. The Technology Enhancement Program is a demonstration-driven effort that includes Demo-I and Demo-II. Demo-II, directed by LTC Erik Mettala, builds on the work of the Autonomous Land Vehicle and research on the NAVLAB at Carnegie Mellon University. Advances in image understanding, neural networks, and computers now make it possible to use a Humvee military vehicle as the research platform, enabling research in the realm of high-performance autonomous navigation. The Demo-II program will transition the results of research developed on Humvee's at CMU, University of Massachusetts, University of

Michigan, Jet Propulsion Laboratory and Martin Marietta Corporation onto the Surrogate vehicle – building four 'Surrogate Semiautonomous Vehicles' to be demonstrated in the fourth quarter of FY 1994. In a recent CMU demonstration, a Humvee operated autonomously in traffic at 63 MPH over a 4 mile stretch of freeway, and ran for 22 miles at more legal speeds.

My predecessor, Rand Waltzman, devoted much time and effort to RADIUS, Research and Development for Image Understanding Systems, a system for the image analyst that focuses on operational intelligence applications. The objective of the study begun in June 1991, is to provide IU technology through research and demonstrations within 5 years, that can be deployed operationally within 10 years. RADIUS is to improve imagery analyst (IA) productivity and the timeliness and quality of exploitation products. The CMEE, based on the SRI Cartographic Modeling Environment, will provide a baseline environment for evaluation by the RADIUS contractor.

The Image Understanding Environment (IUE) was another of Rand's projects. The IUE will facilitiate the exchange of research results within the IU community, and provide a platform for demonstrations and tools for DARPA applications. We expect the IUE to serve as a conceptual standard for IU data models and algorithms. The IUE will facilitate transfer of research results to the applications development community by making available the latest experimental programs in a standardized format and a standardized environment. The design specs for the IUE, a 400 page document, are nearing completion.

With all of this emphasis on applications, I don't want to overlook the crucial IU science base of technical papers. There are 72 technical papers, divided into the following categories: (a) physics-based and low-level vision (12%); (b) stereo (7%); (c) motion (17%); (d) shape recovery and analysis (13%); (e) object recognition and scene analysis (17%); (f) aerial photo matching and interpretation (13%); (g) active vision and visual strategies (5%); robotics applications (12%); and (i) IU software and parallel processing (4%).

Finally, I have to mention funding. In 1990 there were about 80 responses to BAA 90-15 dealing with image research, of which at least 20 deserved funding. Unfortunately, there was money available for only half of these. This points out a basic problem in the field: major university research centers are turning out Ph.D.s who, after a few years of post-doc work, start IU centers

at other universities. Ten years ago there were about half a dozen IU research centers, and now there are at least double that number, with new ones being formed every year. In addition, the cost of such centers continues to increase as more powerful workstations and computers are required to carry out the work. I offer no solution to the funding problem (of course, applications of IU that "knocked people's socks off" would help), I merely point out that there are a lot of empty ricebowls out there and it will take a wide breadth of applications to fill them.

Oscar Firschein, DARPA SISTO
Program Manager
Image Understanding

SECTION I

Principal Investigator Reports

USC IMAGE UNDERSTANDING RESEARCH: 1990-1991

R. Nevatia, K. Price and G. Medioni*
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273

Abstract

This paper summarizes the USC Image Understanding research projects and provides references to more detailed sources of information. Our work has focussed on the topics of 3-D vision (including range data processing, stereo, shape from contour and object recognition), aerial image analysis, motion analysis (including spatio-temporal analysis, 3-D motion estimation, detection of moving objects and an integrated motion system), and parallel processing (including mapping algorithms onto specific or flexible architectures, and processor-time tradeoffs).

1 INTRODUCTION

This paper summarizes our research projects since the last image understanding workshop. Some of this work is described in more detail in other papers in these proceedings [Chung and Nevatia, 1992; Stein and Medioni, 1992; Reinhart and Nevatia, 1992; Ulupınar and Nevatia, 1992; Rom and Medioni, 1992; Kim and Price, 1992; Franzen, 1992; Khokhar and Prasanna, 1992]; this work is covered only briefly in this summary. We also provide references to details for work not described elsewhere in these proceedings. We first give a brief overview of these separate research areas.

1.1 Descriptions from Range Data

We are continuing our work in the acquisition of models from multiple unregistered views.

- In one approach, the raw images are rough-registered using the TOSS system [Stein and Medioni, 1991], then accurately overlapped by the method described in [Chen and Medioni, 1991]. It is then possible to view the integrated data from any viewpoint.
- In an other approach, we instead segment each individual view into simple patches, and describe the data by a graph whose nodes are the patches and whose links are adjacency relations between patches [Fan *et al.*, 1989]. The graphs corresponding to multiple views are matched using a constraint satisfaction network [Parvin and Medioni, 1991b]. The integration proceeds in two steps: first, we generate a composite graph, then intersect surfaces to compute edges and vertices which are part of the B-rep representation.

Finally, given 3D data points, we are investigating the approximation of the underlying surface by a global deformable model. This is achieved by minimizing an energy which expresses the smoothness of the approximation and the closeness to the original data points. Our formalism is an extension to surfaces in 3D of the B-snakes [Menet *et al.*, 1990] for curves.

1.2 Shape Inference and Description from Images

- *Perceptual Grouping:* Most high level vision algorithms, such as shape from contour [Ulupınar and Nevatia, 1988] or line drawing interpretation require perfect data as input, but it is impossible to generate such features with low level algorithms such as edge detectors. Here, we try to bridge this gap by transforming an edge image into a saliency map. We present a non-iterative method based on a field associated with each edge. This field encodes the notions of simplicity, curvature constancy and co-curvilinearity. The results on commonly used test images correspond to "natural" groupings.
- *Description of Planar Shapes:* We study the problem of planar shaped description, and suggest a method to produce an axial representation based on the hierarchical decomposition of the objects into parts. Such a representation is robust with respect to scale, noise and occlusion.
- *3-D Shape Inference from Stereo:* We have been interested in descriptions from stereo images for a long time. Our current focus is on development of a hierarchical stereo system where features are matched at multiple levels of abstraction. Another aspect of this work is in computing object descriptions from

*This research was supported by the Defense Advanced Research Projects Agency under contract F49620-90-C-0078, monitored by the Air Force Office of Scientific Research. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

the stereo matches, which tend to be sparse. We expect our methods to work for objects found indoors as well as man-made objects outdoors.

- *3-D Shape from Contour:* In this project, we are developing techniques for inferring 3-d shape descriptions given only object contours. We have developed a theory that can infer the shape of a class of objects, namely zero-Gaussian curvature surfaces, straight homogeneous generalized cylinders and planar, right, constant cross-section generalized cylinders. One of the recent advances here is extension of our techniques to infer shape of objects made of multiple curved surfaces. We have also started to develop techniques to make our theory work with real images where contours are likely to be fragmented and distracting contours such as markings and shadows present. Our early results are encouraging and described briefly later in this paper.

1.3 Object Recognition

- *Complexity Analysis of the TOSS system:* We have defined a methodology based on efficient coding and hash tables to recognize 3D objects given 3D data, even when the number of models is large [Stein and Medioni, 1991]. We have performed a detailed complexity analysis of the method, which results in $O(n) \leq O_{recognition} \leq O(nm^3)$, where n is the number of matching primitives and m is the number of models in the database. The worst case occurs when the models hypothesized to be in the scene are very similar.

- *Recognition of 3D objects in images:* The more interesting problem is to recognize 3D objects from grey level images. The previous methodology becomes very inefficient, as the number of generated hypotheses increases drastically. We propose instead to generate high level groupings and to use these as matching primitives. The groupings we are using are based on parallel and skew symmetry, U-shapes and closures. Furthermore, we show how to compute these groupings efficiently from segments, and how we keep the number of groupings small. We have obtained encouraging initial results on real images.

- *The "Drop -off" problem:* As an application of our matching methodology, we study the "drop-off" problem, in which an observer is given a topographic map, and is dropped off at an unknown location. We select as matching feature the panoramic horizon curve (corresponding to the sky-ground boundary from a given viewpoint). The polygonal approximation of this curve is compared with precomputed ones using our hash based scheme [Stein and Medioni, 1990]. We have obtained accurate results from real data.

1.4 Motion Analysis

We have a number of projects in the area of motion analysis, with autonomous navigation providing the context for most of the work, though these techniques have a much broader utility.

- *Spatio-temporal Analysis:* Early work in spatio-temporal analysis assumed restricted paths for the moving camera. We developed a slice-based analysis technique that computes dense optical flow estimates for arbitrary observer motion in closely spaced data[Peng, 1991].

- *Motion Estimation:* We have continued our study of multi-frame motion estimation techniques with the development of a system to find the three-dimensional motion and structure estimates for a class of motion, called chronogeneous motion, that includes the standard constant motions and some accelerated motions [Franzen, 1991b; Franzen, 1992].

- *Integrated system for Motion:* In order to use our motion estimation system more fully, we have developed an integrated system that includes hierarchical feature extraction and matching and feedback of the 3-D motion estimation to the feature matching process. This enables the system to tolerate errors and differences in the feature extraction and matching process by removing these inconsistent feature points from the later analysis [Kim and Price, 1992].

- *Mobile Platform:* We use the domain of autonomous navigation to unify our motion work. To this end we have a small project in vision based navigation with a trinocular stereo system for reliable 3-D descriptions of the environment.

1.5 Aerial Image Analysis

Our work in aerial image analysis consists of two major components. First is the transfer of technology funded by DARPA to the RADIUS program. Specifically, we are focussing on the transfer of our techniques for building detection in aerial images [Huertas and Nevatia, 1988a; Mohan and Nevatia, 1989b]. Second project is the continuation of our long range effort of analyzing complex cultural domains. We have chosen large commercial airports as a test domain. In previous work [Huertas *et al.*, 1990a; Huertas *et al.*, 1989a], we have shown good results on the detection of runways and taxiways. In this report, we describe our recent work on aircraft detection.

1.6 Parallel Processing

We are investigating parallel implementations of various vision algorithms developed in our group and elsewhere. We have studied algorithms for stereo and image matching, graph algorithms, sorting on reconfiguration mesh and VLSI architectures for image transforms. In another project, we have studied parallel implementations of various symbolic algorithms. Here, we are interested not only in the efficiency of hardware utilization but also of the programmer efficiency and software maintainability. We find that these goals can be achieved simultaneously by choosing an appropriate architecture. Some of these investigations are described later in this paper.

2 DESCRIPTIONS FROM RANGE IMAGES

Range imagery differs from intensity imagery in that the input directly relates to the geometric *shape* of the objects in the scene. The issues to be addressed are the familiar ones: boundary detection, segmentation, representation, recognition, and pose estimation. How can we represent models to perform matching? In our previous work, we have used multiple views [Fan *et al.*, 1989; Stein and Medioni, 1991; Parvin and Medioni, 1991b], but it is sometimes necessary to also generate an integrated model from multiple unregistered view. The model building procedure to perform the integration at the data level described in [Chen and Medioni, 1991] is non linear and therefore requires an initial guess. We have used the TOSS system [Stein and Medioni, 1991] to provide such a guess. The current limitation is that the function describing the surface of the object must be single valued in some coordinate system. We are working on the segmentation of the object into parts to overcome this limitation.

2.1 Symbolic Level Merging into a B-Rep

The alternative approach consists of generating a symbolic description, such as an attributed graph, for each view, and then merge the different descriptions at this high level. Each view is represented by a graph whose *nodes* are the individual surface patches and the *links* are the relationships between adjacent patches. These patches are inferred from the bounding contours, detected using local operations. We overcome the limitations inherent in this local scheme by forcing the junctions to correspond to possible objects. The implementation is performed using a dynamic network, as explained in [Parvin and Medioni, 1991a]. The rigid transformation between any two views is obtained by matching the graphs describing the views. In our implementation [Parvin and Medioni, 1991b], this is achieved through a two level constraint satisfaction network.

The issue we now address is the integration of multiple views into a single volumetric description. In Geometric Modeling [Requicha, 1980], we construct an object based on geometric operations such as union and intersection of primitive features, i.e. cone, cylinder, etc. This strategy also can be extended to multiple view integration for 3D surfaces, that is, as more area of a given surface becomes visible from other views, its bounding contours are updated to accommodate for the additional information. This approach, however, requires precise knowledge of errors in segmentation and transformation. The errors do not place surfaces at their exact location, when they are transformed to the reference coordinate system. As a result, the merging can become quite complicated. This requires that several tolerances be specified for managing various types of errors. Our approach avoids the low level updating strategy, and concentrates first on building a composite attributed graph of the object. Once the composite graph is constructed, adjacent surfaces are intersected and the location of edges and vertices is computed.

To begin with, the composite graph is initialized to be the attributed graph of a given view. As each additional view reveals more surfaces, they are added to the composite graph. Besides the addition of new links in the composite graph, the union operation updates those attributes that are partially more occluded under one view than the other.

The next issue is surface intersection. For polyhedral scenes, it is rather straightforward, so we only explain the process for curved surfaces. For details, we refer the reader to the technical report [Parvin, 1991].

In each view, a curved surface patch is described by a quadratic equation, and the surface patch is bounded by the corresponding limb and/or orientation discontinuity segments from that view. We adopt the explicit form $F(X,Y,Z)=0$, which is given in matrix form as follows:

$$Q=\begin{bmatrix} A & \frac{D}{2} & \frac{F}{2} & \frac{G}{2} \\ \frac{D}{2} & B & \frac{E}{2} & \frac{H}{2} \\ \frac{F}{2} & \frac{E}{2} & C & \frac{J}{2} \\ \frac{G}{2} & \frac{H}{2} & \frac{J}{2} & K \end{bmatrix}$$

The coefficients of this matrix are approximated by the conventional least square technique, forming the 10-by-10 scatter matrix and searching for the eigenvector corresponding to the minimum eigenvalue. This scatter matrix is constantly updated as data points from other views become visible. These data points are first transformed into the reference viewpoint where their contribution are added to the scatter matrix. Once the scatter matrix contains the contribution from all viewpoints, the coefficients above are estimated for surface description.

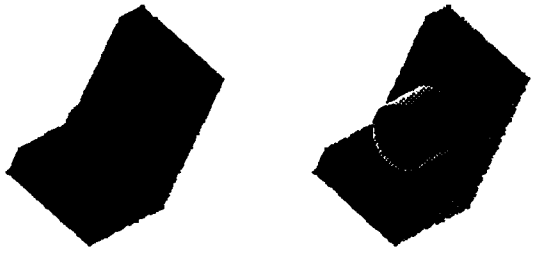
In addition to updating the scatter matrix, the corresponding bounding surface orientation boundaries are transformed into the current viewpoint. Once the composite surface is completed, adjacent surfaces are intersected so that the edge information of the B-rep is computed. In general, surface intersection between quadratic surfaces is a difficult problem. However, additional information maintained in the attribute graph simplifies this task. This includes the list of boundary points along the adjacent surfaces. Of course, this information is not accurate due to segmentation errors, transformation errors and quantization errors. As a result, the surfaces are not placed at the exact boundary location. We propose to use the location of the crease segments as the initial estimate of the surface intersection. Two cases exist for tracing of surface intersection. In both cases, the crease segment dividing the adjacent surfaces can define an auxiliary plane normal to the crease boundary. The first case involves intersection of a quadratic surface and a plane. In this case, the solution to the intersecting curve can be computed in closed form, since the nonlinear system of equation contains two linear equations. The second case involves intersection of two quadratic surfaces. In this case the nonlinear system of equation is solved by Lavenberg-Marquardt algorithm, which is supported by the IMSL package. The basic tracing algorithm is as follows:

Assume that adjacent surfaces are given by $F(X,Y,Z)=0$ and $G(X,Y,Z)=0$ with a point P_i on their intersection. The following procedure will trace the intersecting

boundary:

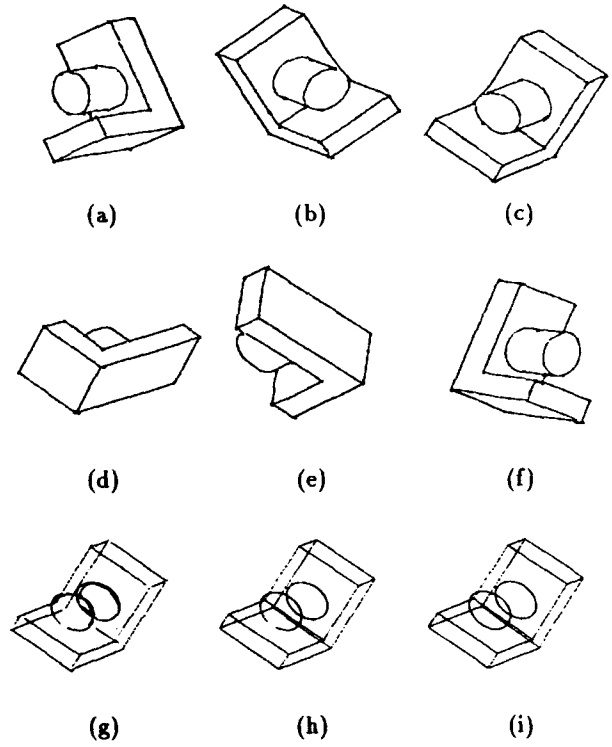
1. compute vector $\vec{d} = \nabla\vec{F} \times \nabla\vec{G}$ at point P_i,
2. estimate $P_{i+1} = P_i + \alpha\vec{d}$ for some small step size α,
3. estimate the auxiliary plane, U, normal to the line P_iP_{i+1} passing through point P_{i+1},
4. refine the estimate of P_{i+1} by solving a nonlinear system of equation given by F, G and U and their Jacobian.

The performance of our system for B-rep generation is illustrated on an example with curved surfaces, as shown in figure 1. Here, 6 views of the object are used to construct the composite graph of the object. These views are matched, the inter view transformation is computed, and the necessary attributes are transformed from one view into the next one. The segmented results for each view of the object, along with the resulting B-rep are shown in figure 2. Note that the union of the object has a gap in the creased curved segment that is completed during surface intersection.



(a) (b)

Figure 1: Intensity and shaded image



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 2: B-rep from multiple views of the object: (a-f) view 1 to 6; (g) union of crease segments; (h) corrected crease segments after surface intersection; (i) B-rep.

2.2 Surface Approximation by Deformable Patches

Overview We present an implementation of deformable models to approximate 3-D surface. It is an extension of our previous work on "B-snakes" [Menet *et al.*, 1990], which approximate curves using B-splines. The user provides an initial simple surface, such as a cylinder or a sphere, which is subject to internal forces (describing implicit continuity properties such as tension and bending) and external forces which attracts it toward features. The problem is cast in terms of energy minimization, and is solved iteratively. our choice of basis functions leads to reasonable complexity and good numerical stability. We show results on real range images to illustrate the applicability of our approach. The advantages of this approach are that it provides a compact representation of the approximated data, it gives a C_1 continuous (for quadratic spline) analytical description of the data, which allows computation of differential properties, and tends itself to application such as non rigid motion tracking and object recognition.

Surface Fitting The idea of fitting data by a deformable model can be found in the work of Kass *et al.*[Kass *et al.*, 1987] in 2D. Such models were generalized in 3D by the same authors [Terzopoulos *et al.*, 1987; Terzopoulos, 1988] for a surface of revolution. More recently, Cohen [Cohen *et al.*, 1991], Terzopoulos [Terzopoulos and Metaxas, 1991] and Pentland[Pentland and Sclaroff, 1991] have formulated new methods for 3D data. Our approach is closely related to [Cohen *et al.*, 1991], except that we use finite differences as opposed to a finite element method. The formalism we are about to establish amounts to deforming the initial surface to conform as closely as possible to the given 3D data points. This is achieved by defining an attraction force field around these points to bring the initial surface closer to them, and is solved by introducing an energy dissipation functional to dissipate the kinetic energy during the motion. In our case, since the entire surface is defined in terms of control points, the positions of all control points get updated at each iteration. We are given an initial surface (such as a cylinder), defined with MN control vertices. Any point on the surface is defined by the patch (i, j) to which it belongs $(0 \leq i < M, 0 \leq j < N)$ and its coordinate (u, v) on the patch $(0 \leq u, v \leq 1)$. Therefore we write the position $(V_x, V_y, V_z)^T$ of each point as $\vec{V}(i, j, u, v)$. The total energy of the surface is the sum

of the energies at each point.

$$E_{total}=\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\int_0^1\int_0^1[|E_{int}(V(u,v))|+|E_{ext}(V(u,v))|]dudv$$

The internal energy expresses the smoothness of the first and second derivatives of $\vec{V}$, and can be written as

$$E_{int} = W_\alpha[|\frac{\partial\vec{V}}{\partial u}|+|\frac{\partial\vec{V}}{\partial v}|]+W_\beta[|\frac{\partial^2\vec{V}}{\partial^2 u}|+|\frac{\partial^2\vec{V}}{\partial u\partial v}|+|\frac{\partial^2\vec{V}}{\partial^2 v}|]$$

The external energy is a potential energy which attracts the surface toward the data points. In our implementation, we precompute in a digitized volume (typically $64 \times 64 \times 64$) the distance of each voxel to the closest data point. The distance from a point on the approximating surface is simply that of the voxel the point falls in. The energy for each patch is computed at $K \times K$ equally sampled points on the surface, therefore E_{ext} is

$$E_{ext} \approx \frac{area(i,j)}{K^2} \times W_{ext} \times \sum_{k_1=0}^{K-1}\sum_{k_2=0}^{K-1} G(F(\frac{k_1}{K},\frac{k_2}{K}))$$

and $area(i,j)$ is approximated by the sum of eight triangles.

For a quadratic B-spline surface, a control point $\vec{C}(i,j)$ only affects 9 patches $P(k,l)(i-1 \le k \le i+1, j-1 \le l \le j+1)$. So, to minimize the energy with respect to the position of each control point, we only consider these patches. For the X component, we have the following equation:

$$\frac{\partial E_{total}}{\partial X(i,j)} = \sum_{k=i-1}^{i+1}\sum_{l=j-1}^{j+1}\frac{\partial E(k,l)}{\partial X(i,j)} = 0$$

For a quadratic spline, this expression is a function of 25 control points $\vec{C}(k,l), i-2 \le k \le i+2, j-2 \le l \le j+2$, so we end up, for each control point $\vec{C}(i,j)$,with an equation of the type

$$\sum_{h=i-2}^{i+2}\sum_{k=j-2}^{j+2} C_{hk}\cdot X(h,k) + D(i,j) = 0$$

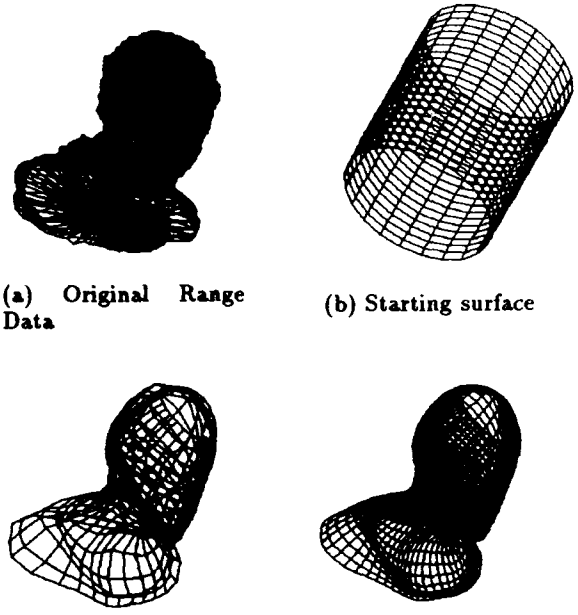
This can be expressed in matrix form as $AX + D = 0$, where A is a MN by MN matrix, X and D are MN by 1 matrices. This system of equations is solved iteratively

$$X_{t+1} = (A+\gamma I)^{-1}\cdot(\gamma X_t - D(x,y,z)),$$

where γ is the Euler step size.

Results We present an example with real data to demonstrate our approach. The data is a 360 ° laser range finder image of a human head scanned by Visual Computing Group (courtesy of Vision and Modeling group, the Media Laboratory, MIT). There are about 2500 points, as shown in figure 3(a). The original surface is defined by 22×22 control points equally spaced on a cylinder, as shown in Figure 3(b). The final position of the control points is shown in figure 3(c) after 25 iterations, and the corresponding surface is displayed in Figure 3(d). The running time for this example is 30 minutes on a Sparc 2 workstation. We used the following parameters:

$\gamma = 0.05, W_\alpha = 0.001, W_\beta = 0.05, W_{ext} = 1.0$



(a) Original Range Data

(b) Starting surface

(c) Final position of control points

(d) Final surface

Figure 3: Fitting a B-spline surface

Future This method is a direct extension to 3D of our work on B-snakes. We are interested in reducing the computing time by an order of magnitude, and believe that this can be achieved using an adaptive approach.

3 SHAPE INFERENCE AND DESCRIPTION FROM IMAGES

3.1 Perceptual Grouping

An area which is likely to improve results in computer vision is the one of perceptual grouping. This area can be classified as a mid-level issue directed toward closing the gap between what is produced by state-of-the-art low-level algorithms and what is desired as input to high level algorithms. Many researchers resort to using synthetic data as their input because of that weakness. Figure 4(a) depicts an example of perceptual grouping easily experienced by the human visual system. The circle in the middle is easily distinguishable from its noisy background. Furthermore, we tend to fill the gaps and accept the fragmented circle as a complete one. Figure 4(b) shows a more complex case with overlapping features. A striking example, often used in the psychology literature, is that of the Kanizsa illusion (Figure 4(c))[Kanizsa, 1976]. Here we perceive edges which have no physical support whatsoever in the original signal. Lowe [Lowe, 1987] discusses the Gestalt notions of co-linearity, co-curvature and simplicity as important in perceptual grouping. Ullman [Sha'ashua and Ullman, 1988] suggests the use of a saliency measure to guide the grouping process, and to eliminate erroneous features in the image. The approach described here is closely re-

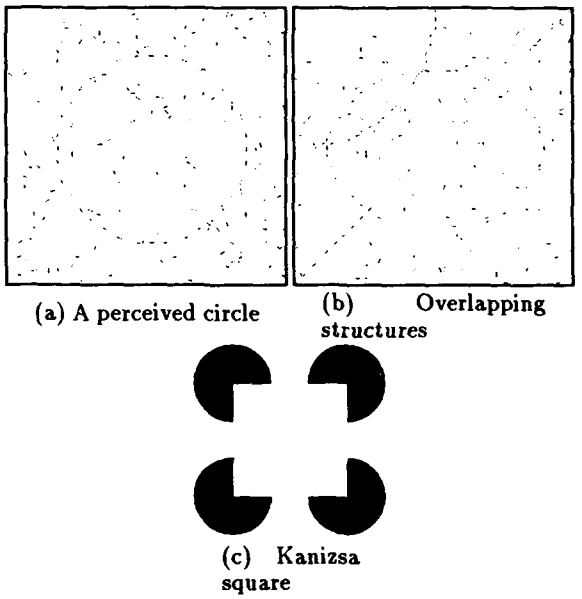


Figure 4: Examples of perceptual groupings

lated to Ullman's, in the sense that a saliency map is first constructed from an edge image, and later higher-level features are referred. The proposed approach is capable of "highlighting" structures which are salient, as well as interpolating gaps in a smooth manner, and remove noisy edgels in a given image, all in a unified non-iterative scheme. The underlying goal is to keep the interpretation as simple as possible in the "Gestalt" sense. This translates into three major constrains:

1. Co-curvilinearity In the lack of other cues, continuation is the only interpretation, and so is co-curvilinearity.
2. Constancy of curvature We tend to extend a curve of some constant curvature with the same curvature.
3. Favoring low curvatures over large ones Humans seem to connect fragmented line segments in a way that the increase in total curvature is minimum (see Ullman [Sha'ashua and Ullman, 1988]).

With that in mind, we have devised a technique that implicitly imposes the above constrains in the form of an *Extension field* emanating from each edge segment, as described next. The calculation complexity of the saliency map is linear in the number of edge elements in the image and thus bounded by the size of the given image. Ideally, a saliency map should assign large values of probability along these illusory lines, and also specify a direction of most probable continuation of the segment. We will show how our scheme treats examples of such nature.

Extension Fields An Extension field is a non-normalized probability directional field describing the contribution of a single edge element to its environment in term of length and direction. Put other words, it votes on the preferred direction and the probability of existence of every point to share a curve with the original segment.

The field is of infinite extent, although in practice it disappears at a predefined distance from the edge. Figure 5 depicts such a field. Since we favor large and constant curvature, field direction at a given point in space is chosen to be tangent to the circle passing through the edge segment and that point, while its strength is proportional to the radius of that circle. Also, the strength decays square-exponentially with the distance from the origin (the edge segment).

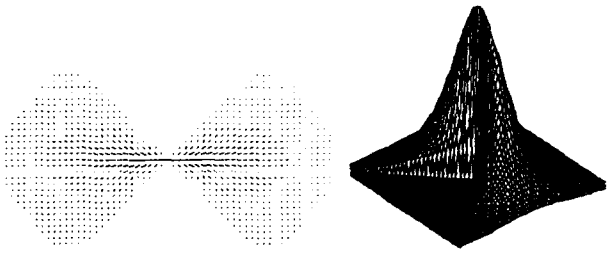


Figure 5: The elementary extension field

$$f(x,y) = \begin{cases} r(x,y)e^{i\theta(x,y)} & \text{if } |y| < |x| \\ 0 & \text{otherwise} \end{cases}$$

where $\theta(x,y) = tan^{-1}(\frac{2xy}{x^2-y^2})$,

and $r(x,y) = Ae^{-B\frac{x^2+y^2}{2|y|}} \cdot e^{-CD^2}$

and D is the arc length along the circle linking the origin to (x,y).

The assignment of actual probabilities to the field is performed as follows. We consider two short edge segments, perpendicular to each other and apart. We assign probabilities to the field in such a way that all paths connecting these points get roughly the same saliency, such that any one best path does not exist between the two. This seems to be in agreement with human perception.

The uncertainty with regard to where an isolated edge should be extended (if at all) is very high to begin with. When several co-linear edge segments are combined, the resulting field becomes more and more directed and the uncertainty grows smaller. For instance, when two co-linear line segment fields are combined, the probability for that longer segment to extend to a very curved line becomes smaller, while the weights in the direction of the lines grow larger. The same intuitive result is produced when two segments form an obtuse angle. Furthermore, the gaps are bridged in a smooth fashion. The whole process can be thought of as a directional convolution with the above field (mask). The resulting map is then a superposition of a collection of fields each oriented along a corresponding short segment.

Combining individual field elements is of great importance. Ideally, we would want an averaged majority vote regarding the preferred orientation of a given position. In practice we used a pair-wise vector addition of field elements. The addition favors the direction which makes

the resulting vector the largest. In tie situations (2 vectors are perpendicular) an arbitrary choice is made.

The whole scheme is resolution dependent. This means that the actual "primitive" line segments are arbitrarily defined to have a specific length. From this point and on, they are considered unit length. It is thus desired in general to process an image starting with a coarse resolution to extract large features, and then go on to finer features. In our current system, only the coarse resolution is considered.

It is interesting to note that, although the process is local in essence, a global percept emerges as a result.

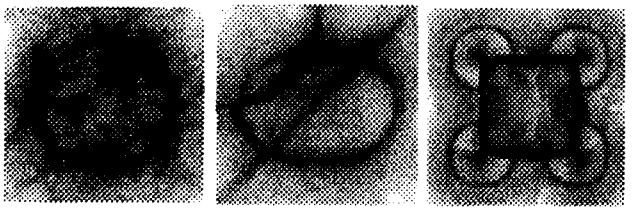
The addition of random noise to an image is expected to create a distributed map of votes, and thus not to interfere with truly salient patterns. When an accidental formation of random segments does give rise to high values in the map, that formation is perceived as significant to humans as well.

In most cases, applying the convolution once is sufficient to achieve a meaningful saliency map. However, in situations where the noise levels are particularly large, we threshold the saliency map and recalculate the convolution on that map.

Extraction of high-level features Once a saliency map is acquired, a process which iteratively 'removes' salient group is started. This process first removes the most salient group, recalculates the saliency map, and then proceeds to remove the next most salient group. The complexity of this process is thus proportional to the number of features in the image. It is guaranteed to terminate since at each iteration the overall power of the field is strictly reduced by removing a feature.

We use a directional roof-top following algorithm. The linking process starts at the point of largest saliency and advances in the general direction dictated by the orientation of the current position.

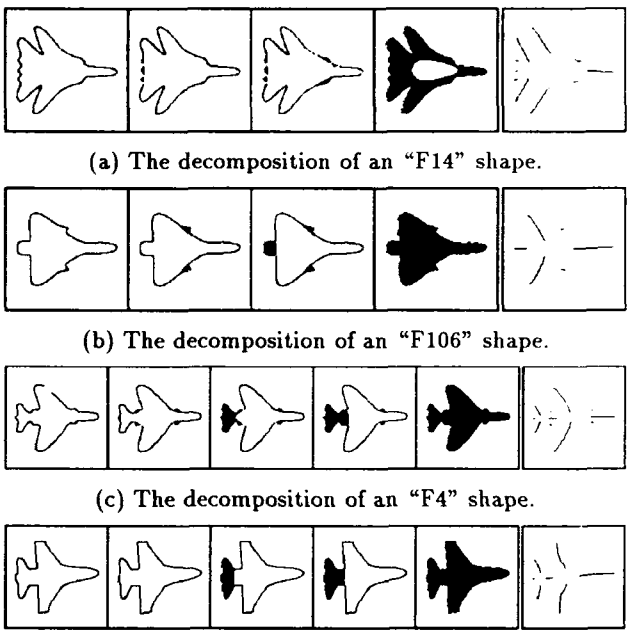
Results We have tested our approach on the three examples shown in Figure 4. The saliency map produced is shown (strength only) as an intensity image (Figure 6). The result of following the path of highest saliency produces the expected groupings.



(a) (b) (c)

Figure 6: Intensity of saliency maps

Conclusions and future directions The above scheme was shown to achieve useful results for some subset of problems in perceptual grouping. We intend to incorporate two additional kinds of fields. The first will enhance perpendicular relations between lines, such as junctions and edges which perceptually terminate at a vertex. The second will enhance parallel relations in the image. When all these fields are combined, we expect to get a much more powerful saliency measure.



(a) The decomposition of an "F14" shape.

(b) The decomposition of an "F106" shape.

(c) The decomposition of an "F4" shape.

(d) The decomposition of an "F16" shape.

Figure 7: Decompositions obtained from real data.

3.2 Planar Shape Description and Decomposition

Shape description is a major problem in machine perception, and is the basis for recognition. Many approaches have been suggested, but none provide a complete and natural solution. We have developed a method for producing an axial representation of a shape based on a hierarchical decomposition of the shape into its parts. The novelty of our approach lies in the combination of several competing approaches and tools, into a unified scheme and an efficient implementation producing natural descriptions. The details are given elsewhere in these proceedings [Rom and Medioni, 1992], and we have obtained natural decompositions on shapes obtained from real data, as shown in Figure 7.

3.3 3-D Shape Inference from Stereo

Use of stereo is common to recover 3-D structure of the scene by using multiple images. Traditionally, the major problem in stereo has been considered to be that of finding correspondences, *i.e.* finding the images of the same physical entity in the multiple images. It has been thought that surface description and segmentation processes follow, and are largely independent of, the correspondence problem. We argue that this view is not completely correct. While finding a good solution to the correspondence problem is critical, we can typically only expect to get a sparse depth map from this step. Thus, finding surface descriptions still requires grouping operations. We argue that the grouping operations may, in

fact, help the correspondence process and in this sense the two are not totally independent.

We have been developing a stereo system that constructs a hierarchy of descriptions in each image and matches these descriptions. Our first experience with this approach was in description of buildings that are composed of rectangular structures [Mohan and Nevatia, 1989b] from stereo pair of aerial images. Our current system uses the following feature groupings: edges, curves, junctions and ribbons. Matches are made at the highest available level first and propagated down to lower levels. A block diagram of this system is shown in figure 8. A major component of this method consists of a careful analysis of occlusion properties in stereo which gives strong constraints on matching of junctions of contours and allows us to distinguish limb boundaries from surface discontinuity boundaries (creases). This system is new since our last progress report, however, it has been published in open literature elsewhere [Chung and Nevatia, 1991], so to save space we will omit the details here.

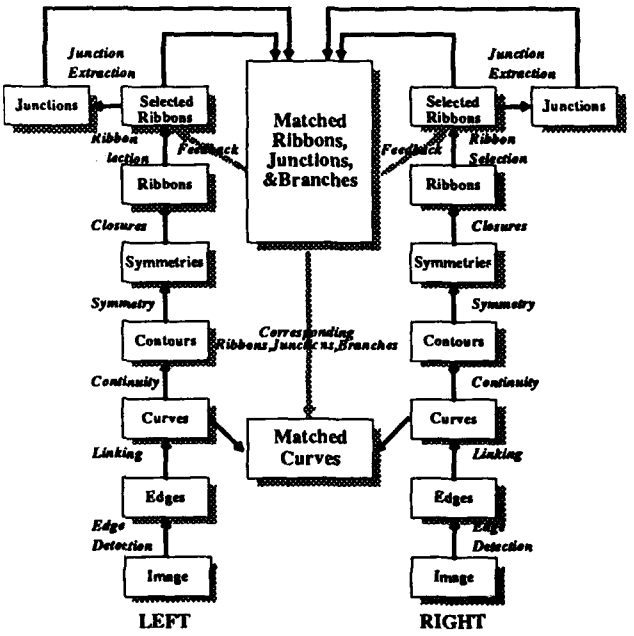
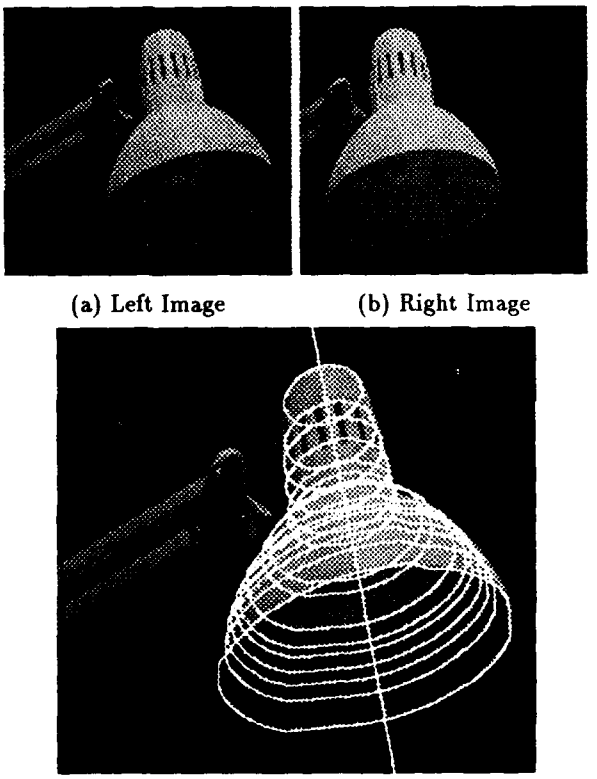


Figure 8: Overview of our stereo system.

One application of this stereo system we are investigating is shape description of Straight Homogeneous Generalized Cylinders (SHGCs). Many curved objects, including SHGCs, produce boundaries where significant sections are view-point dependent (limb) boundaries. These boundaries can not be matched in the two views without producing significant errors. Worse yet, incorrectly segmenting the surface at these boundaries produces a poor description. This problem was first addressed by Lim and Binford [Lim and Binford, 1988]. Their system makes some assumptions that are not always correct and they provided no methods for separating the limb boundaries from the other boundaries. We have developed a method that is mathematically sound and also uses our junction analysis technique to identify the limbs. This method is described in detail elsewhere in these proceedings [Chung and Nevatia, 1992]. Results fora a real examlple using this method is shown in figure 9.



(a) Left Image (b) Right Image

(c) Recovered volumetric description

Figure 9: Results of hierarchical stereo matching and volumetric shape recovery for a scene of a lamp

3.4 3-D Shape from Contour

In some cases, only a single image of a scene is available and we need to infer the 3-D structure from this single image. Humans are usually very good at this task, but the process we use is not well-understood. Of the many cues that are available, such as shading, shadows and texture, we believe that the most significant one is the shape of the 2-D contours themselves. Inferring 3-D surface from 2-D contours is a well-known and difficult problem. We have developed a theory for this that we believe is applicable to a large class of scenes that we encounter and that yields results in agreement with the human perception.

A set of 2-D contours can, of course, arise from infinitely many 3-D shapes, though we typically perceive only one (or two). To make such inferences, some assumptions about the nature of the scene must be made. Our objective in constructing our theory was to minimize the number of assumptions made and to generate results that agree with human perception. (In a certain sense, the only test for shape from contour algorithms

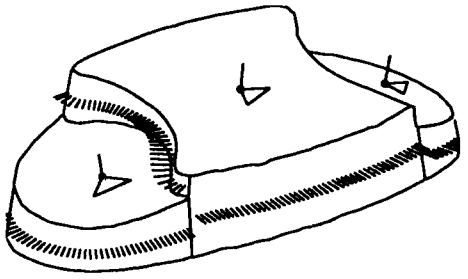


Figure 10: The needle image obtained from the computed surface normals.

can be comparison with human perception, since even in those cases where the 3-D object projecting the 2-D shape is known to us, the same 2-D contour *could* be produced by another 3-D object as well.)

Our technique is based on observing certain kinds of symmetry relations among image curves. We show mathematically the conditions under which certain classes of surfaces exhibit these symmetries. Though the reverse is not always guaranteed, we assume that the symmetries do indicate the presence of the corresponding classes of surfaces. Further, the contours and the symmetries allow us to formulate some constraints on the quantitative shape of the surfaces being viewed. The constraints that derive purely from the geometry of the surface are, however, not sufficient to compute the precise shape of the surface and leave some degrees of freedom unspecified. These remaining degrees of freedom are fixed by using some simple perceptual properties.

Our approach has so far been developed for the following classes of surfaces:

- Zero-Gaussian Curvature (ZGC) surfaces
- Straight Homogeneous Generalized Cylinders (SHGCs)
- Planar, Right, Constant Cross-section Generalized Cylinders (PRCGCs)

We believe that this class of objects is broad and covers a wide range of objects, particularly in man-made environments. In previous papers [Ulupınar and Nevatia, 1990a; Ulupınar and Nevatia, 1990b; Ulupınar and Nevatia, 1991] we have described these methods in detail and shown results on a number of examples. In recent work, we have extended our technique to work on objects consisting of several curved surfaces (restricted to be zero-Gaussian curvature surfaces). Our perception of a surface can be heavily influenced by the perception of the neighboring surfaces. Our technique recovers shape of all the visible surfaces of an object simultaneously. Figure 10 illustrates the type of results for this technique. Elsewhere in these proceedings [Ulupınar and Nevatia, 1992], we describe the details of this technique.

3.5 Working with Imperfect Data

In these previous papers, however, we assumed that the input to our method consisted of clean and complete line drawings. That is, we assumed that we were given complete object boundaries (without gaps) and that no other boundaries, such as those that may be caused by surface markings, shadows and noise were present. This is, of course, unrealistic when boundaries are derived from real images not necessarily taken under highly controlled conditions. Our current work aims to address these difficulties. This is work in progress, but we have already gotten some very encouraging results. We briefly describe our approach and some results here.

Our approach is based on perceptual grouping. Normally, perceptual grouping uses criteria such as proximity, continuity, symmetry and closure. While the importance of such criteria is widely accepted, the precise properties for implementation are not clear. In past work [Mohan and Nevatia, 1989a], we showed promising results using intuitive definitions. However, for applications to the current task, we have a major advantage- the precise properties that allow us to infer a ZGC, SHGC or PRCGC, the only classes of surfaces for which we know how to infer 3-D structure, are known to us. Thus, our grouping can be to search for the presence of these specific properties. This not only helps us avoid the computational complexity of the general grouping approaches, but gives groups that we know how to interpret as 3-D objects. In other words, the process of grouping and shape description are combined.

Our grouping consists of several steps (see figure 11).

1. First step consists of grouping curves based on cocurvilinearity, we use the method given in [Mohan and Nevatia, 1989a]. This step can bridge short breaks.
2. Next step attempts to bridge large breaks and to detect missing terminations and discontinuities of curves. For this, we use the symmetry properties themselves to aid in the grouping. This process consists of the following components:
   (a) Initially, we fit B-splines to the given curves and find symmetries among these using the method of Saint-Marc [Saint-Marc and Medioni, 1990]. This method gives sparse, local correspondences and is unable to distinguish between "desired" symmetries and others, which can only be done by more global properties.
   (b) Local symmetries are grouped into more global symmetries. Our grouping criteria account for different ways two parallel symmetric curves can be broken due to occlusion, low contrasts or imperfections in the edge detection. An example is shown in figure 12(a). Competing hypotheses yield alternative groupings which are evaluated at higher levels where more global information becomes available (see figure 11).
   (c) The above step can give symmetries that are not complete. Next we hypothesize possible terminations, based on continuity of one of the curves giving the unterminated symmetry and other contours in the image to give closure. An example is shown in figure 12(b).
3. Verification of global symmetries: here we test for grouped symmetries to verify global regularity crite-

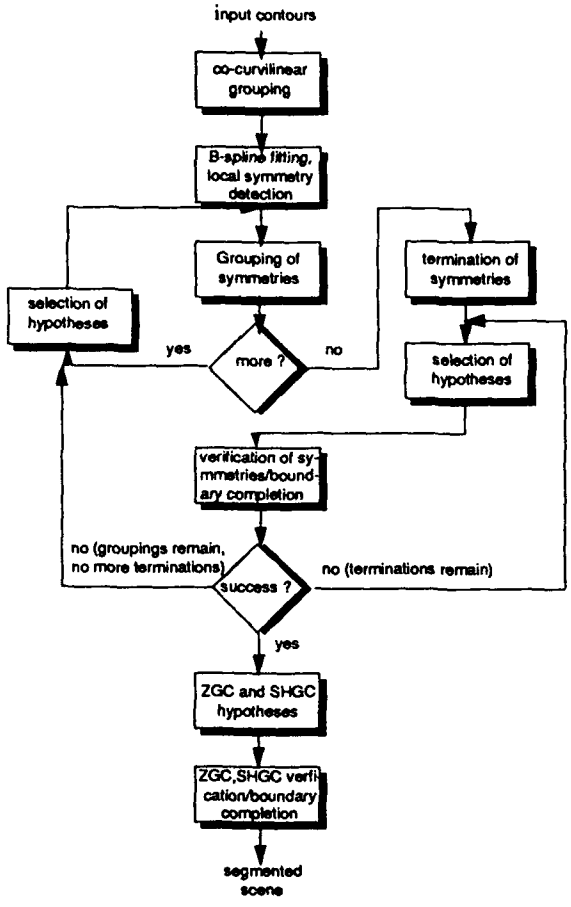


Figure 11: Block Diagram of the method

ria that consist of preferring monotonic, linear correspondences with closed boundaries. Note that this step will remove the effects of surface markings, even if they happened to form symmetries accidentally in step 2. Once consistent global symmetries and, consequently, global curves are obtained, large gaps can be filled in in a way dictated by the similarity between symmetric curves.

4. In the final step, global symmetries and completed boundaries together with properties of ZGCs and SHGCs are used to hypothesize existence of such surfaces in the scene. Here, complex cases of terminations (not handled in step 2.c above) and limb boundary completion can be handled.

We show the operation of our method by three examples, one is an LSHGC (ZGC surface), the other two are SHGCs.

Figure 13(a) shows the boundaries of an LSHGC with many breaks and extraneous boundaries. Figure 13(b) shows the B-spline fit on the grouped curves and the local symmetries detected (their axes) (output of steps 1 and 2.a above). Figure 13(c) shows completed symmetries and boundaries. This input is now in a form that can be handled by our shape from contour system.

Figure 14(a) shows the boundaries of an SHGC. Notice the large break. Figure 14(b) shows the B-splines fit and the local symmetries. The gap has been filled in figure 14(c) as a consequence of the global symmetry existing between the top cross-section boundaries and the bottom ones; i.e. the bottom part suggests how to complete the missing top part. Figure 14(d) shows the visible cross-sections and the recovered axis. For this, we use the property that lines of symmetry intersect on the axis.

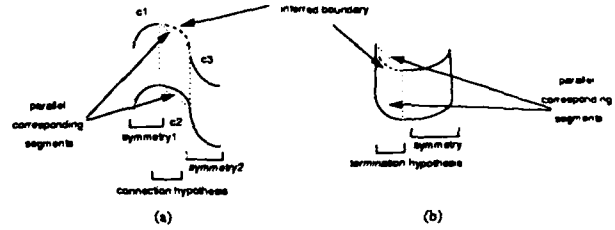


Figure 12: Connection (a) and termination (b) hypotheses and boundary inference

Figure 15(a) shows the boundaries obtained from another SHGC. Note that this time part of the right limb boundary is missing. This is a more serious situation than missing part of the cross-section boundary which can be bridged by using the parallel symmetry of the cross-sections. The limb completion is done after the cross-sections in the incomplete part of the SHGC have been recovered. We use the property that limb boundaries and cross-sections are tangential. Limb points are points on these cross-sections defining a tangential envelope.

Figure 15(d) shows the axis, recovered cross-sections and completed boundaries which can now be passed to our shape from contour system.



Figure 13: Processing of an LSHGC contour

These examples are meant to illustrate how the powerful constraints given by our shape from contour theory can be used to infer shapes even in presence of significant breaks and markings. This system is still in development and has not been fully tested. In the future, we intend to develop methods that work on scenes with multiple, occluding objects and with complex objects that consist of assemblies of primitives we have studied.

4 OBJECT RECOGNITION

4.1 Analysis of the TOSS system

We have introduced the TOSS object recognition system which is able to match general three dimensional

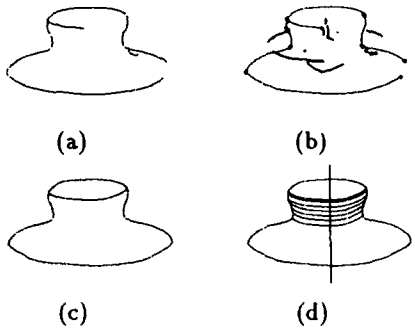


Figure 14: Application on an SHGC contour

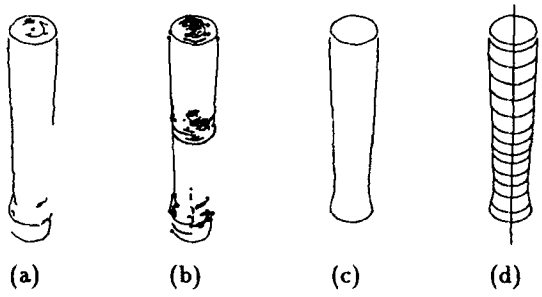


Figure 15: Application of the method to a SHGC contour with cross-section curve and limb broken

objects from partial 3D data in an efficient way by using a method called *structural indexing* [Stein and Medioni, 1991]. Our algorithm uses a combined representation, which captures information about both smooth patches and discontinuity lines: For some objects, such as polyhedra, it is natural to use a representation based on edges. For that reason, we use as basic feature for the representation of surface and depth discontinuities, the *3D curve*. We achieve a robust and stable representation by using multiple line fitting tolerances to obtain a set of polygonal approximations. The polygonal approximations are grouped in sets of connected segments. These super segments are encoded based on the angles between consecutive segments, providing invariance with respect to rotation, translation, and scale (even though scale is not needed in three dimensional object recognition).

For some objects, however, such as objects bounded by free form surfaces, it is difficult to use edges for the representation. Therefore, we also use *splashes*, based on small surface patches where we can compute differential properties in a reliable way. A splash consists of a radial grouping of surface normals. It is a local Gaussian map describing the distribution of surface orientation along a geodesic circle. A splash can be represented by two two-dimensional periodic functions, which can also be combined into one, compact, three dimensional curve. This allows us to use a unified representation scheme for the splash *and* the above mentioned 3D curve.

We illustrate the recognition of three models of composers (shown in figure 16) in a scene with occlusion.

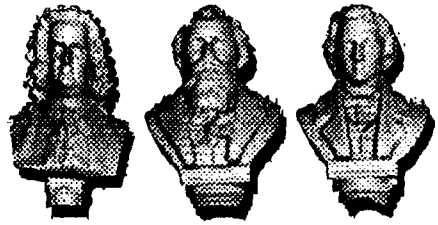


Figure 16: Three Composers

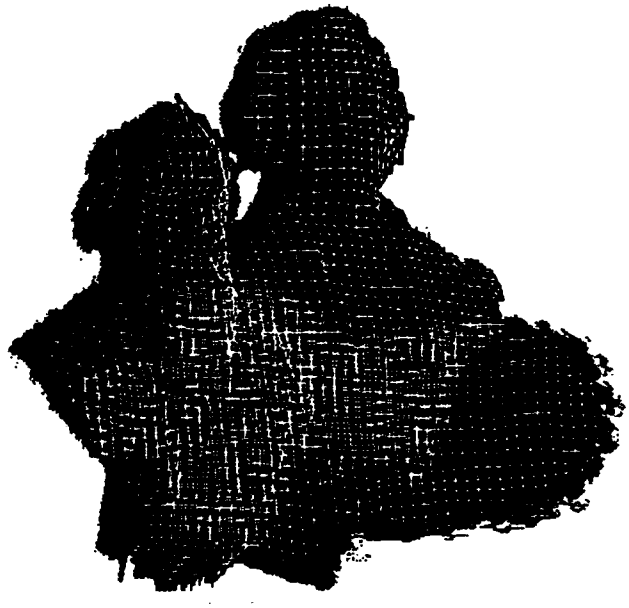
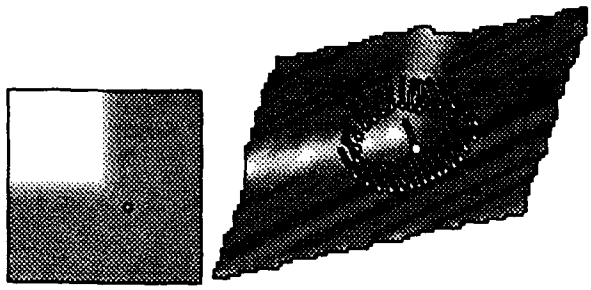


Figure 17: Projection of Detected Busts in Scene 3

Figure 17 shows the reprojection of the recognized objects onto the scene.

Robustness and Stability In order for our representation to be useful in recognizing objects from real data, it is necessary to address the issues of robustness and stability of the splash representation in the presence of noise. In particular, we must examine the robustness of a splash regarding location uncertainty, study the effects on representation and matching of an error in the reference normal orientation, and ask how much noise can be added to the surface patch and still have a stable representation.

We have found empirically that our representation is adequate, since our system performs well on real data. We would have liked to model the behavior of our scheme on arbitrary free form surfaces, but they are too general to be of any help. Instead, we present a full analysis on a simple analytic model of a specific surface patch, as shown in Figure 18(a) and (b), for all three issues (location, orientation, and surface stability). Furthermore for the orientation robustness issue, we study the effects of noise in the general case of a 3D curve (representing



(a) Location of the Reference Splash

(b) Qualitative Side View

Figure 18: The Corner Surface

either one of our features).

Robustness in the Location To answer the question of how robust the representation of a splash is with respect to location accuracy, we use empirical data based on an example environment as described above.

For small quantizations, the patch is recognized only if the splash is very close to the location of the original splash, whereas larger quantizations allow a much larger latitude in location uncertainty. Of course, if the quantization is too large, the splash may be confused with splashes from other locations. We therefore observe the desired robustness with respect to location uncertainty for this surface patch for the some of the quantizations.

Robustness in the Reference Normal Here, we examine the influence of noise in the reference normal on the representation of a splash. Adding an error angle ϵ in the direction δ to the reference normal, how is the curve $\vec{v}(\theta) = \begin{pmatrix} \phi(\theta) \\ \psi(\theta) \end{pmatrix}$ influenced?

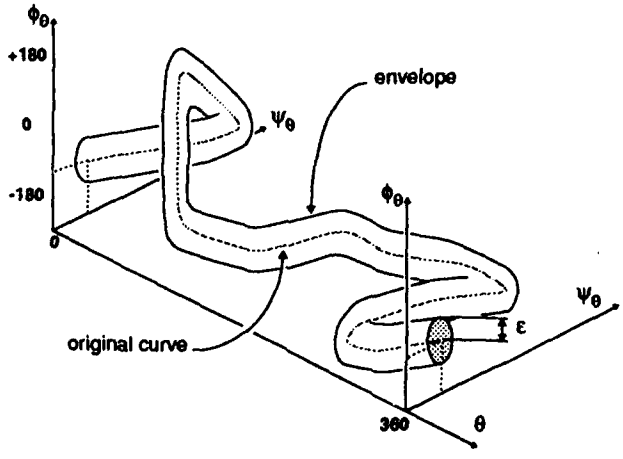


Figure 19: Envelope of a 3-D curve

We find that the maximal error for $\Delta\phi$ is bounded by:

$$\|\Delta\phi\| = \|\phi - \phi'\| \leq \epsilon$$

and that a bound for the maximal error for $\Delta\psi$ as

$$\|\Delta\psi\| = \|\psi - \psi'\| \leq \epsilon.$$

We can now address the problem of how the representation might vary. Instead of a single curve $\vec{v}(\theta)$ we have now to deal with an envelope of curves $\vec{V}(\theta, \varepsilon)$ which is a disc of radius ε swept along $\vec{v}$ (see Figure 19).

What are the polygonal approximations that approximate such a set of curves? How are the angles between consecutive line segments affected? To make a general statement is very difficult. Therefore we make some simplifying assumptions. As a result, we find that, for small quantizations, the patch is recognized only if the reference normal is very similar to the reference normal of the original splash. An uncertainty of up to two degrees is bearable and only for a certain δ direction. Larger quantizations allow a much larger error in the reference normal, and more freedom in the δ direction. The shape of the different distributions with respect to δ is dependent on the underlying surface, and the quantization.

Stability with respect to Noise We want to address the question of how corrupted can the underlying surface be to reduce the stability of the splash representation. We add zero mean Gaussian noise $a\mathcal{G}(0, \sigma)$ with a the amplitude (in pixels) and σ the standard deviation, to the surface s to get a corrupted surface, and compute the splash at the same location as the reference splash on the corrupted surface. For a given quantization we then match the "corrupted" splash against the reference splash. We find that the larger the quantization, the more stable is the splash representation, the smaller the quantization, the less stability we can observe. The matching behavior is not affected by either increasing the amplitude of the noise and keeping the standard deviation small or decreasing the amplitude and having a large noise deviation.

Complexity Analysis The whole issue of complexity focuses on the question: "What is the discriminative power of the features in the system?" The answer to this question is made up of two parts. The first part is the analysis of the retrieval process and the number of generated hypotheses. The second part is the discussion of the verification step and the cost of grouping them into consistent clusters. When we talk in the following about "scene features" we consider only the scene features, which lead to the generation of at least one hypothesis. For the complexity analysis, we ignore the scene features whose code is disjunct from all the keys in the hash table (they also do not add any cost in practice). Before we start to answer the above question we define the parameters involved:

- n = the number of features in the scene,
- d = the number of features per table entry,
- m_s = the number of models in the scene,
- and m = the number of models in the data base.

To simplify the discussion, we make the assumptions that every model has the same number of super segments, and that the entries are equally distributed over the table ($d = const$). Furthermore we assume that every model consists of f features ($f = const$).

Hypotheses Generation When we have n features in the scene, it is obvious that the cost to generate all candidate hypotheses with indexing is $O(n)$. The important issue for the cost of the complete recognition is *not* the cost of retrieval, but h, the number of hypotheses generated, because this number has a crucial effect on the verification step. The number h is proportional to the number of features per table entry d ($h = d \cdot n$). The larger h, the slower is the clustering into consistent clusters in the following verification. Therefore we are interested in a small h, which corresponds to a small d. This, so called "ability do discriminate", is influenced by the amount of noise in the data, the amount of similarity between models, and the grid size for the interest operator.

To summarize, the crucial value for the retrieval stage is h, the number of generated hypotheses. In a best case, this number is very low. The theoretical best case is, when every hypotheses votes for a different model ($d = 1$ and $n = m_s$). This results in $h = n$. The worst case corresponds to a large value of h. There is only little discriminative power in the features. This means that most features are encoded with the same code (large d). Every scene feature generates all possible candidate hypotheses and therefore the overall number of retrieved hypotheses candidates is $h = f \cdot m \cdot n$. This is the approach taken by many systems of the past, which use either points or lines as basic features, and must therefore perform the discrimination task during the verification step.

Verification As mentioned above, the task of the verification is to cluster the h hypotheses into mutually consistent clusters. This is done for every entry in the correspondence table.

Best Case: In the best case, there is a lot of discriminative power in the features. This corresponds to a low d value. The $h = d \cdot n$ hypotheses are divided in the correspondence table according to which model the hypothesis votes for. For m_s models in the scene we have m_s entries in the correspondence table with $\frac{h}{m_s}$ hypotheses each. For every entry these $\frac{h}{m_s}$ hypotheses have to be grouped with respect to the geometrical constraints. We distinguish the clustering process based on different cases:

- *Every model in the scene occurs only once*: In this case, the hypotheses in one entry either vote for the model instance or they are wrong hypotheses. This leads to

$$O_{best}(n) = m_s \cdot O(\frac{h}{m_s}) = O(d \cdot n) = O(n).$$

- *Every model in the scene can occur more than once*: In this case the clustering of the hypotheses in one entry cannot be done in $O(\frac{h}{m_s})$, but $O(\frac{h^2}{m_s^2})$ instead, leading to a complexity of

$$O_{best}(n) = m_s \cdot O(\frac{h^2}{m_s^2}) = O(\frac{d^2 \cdot n^2}{m_s}) = O(n^2).$$

The best case has a noteworthy side effect. Assuming the number of scene features n fixed, and examining the complexity O_m with respect to the models in the data base, we find that it grows as $O_m = O(k \cdot m)$ when m is the number of stored models, and $k < 1$.

Worst Case: In the worst case, there is little discriminative power in the features. This corresponds to a high value for d. The overall number of retrieved hypotheses candidates is $h = d \cdot n$ with $d = m \cdot f$. These h hypotheses are divided in the correspondence table according to which model the hypothesis votes for. For m models in the data base we get, in the worst case, m entries in the correspondence table with h hypotheses each. For every entry, these h hypotheses have to be grouped with respect to the geometrical constraints. Clustering these h hypotheses results in a complexity of

$$O_{worst}(n, m) = m \cdot O(h^2) = O(f^2 \cdot n^2 \cdot m^3) = O(n^2 \cdot m^3).$$

In the worst case, the ratio of good versus bad hypotheses is very small.

Summary As a conclusion we get the result, that the practical complexity of our system is

$$O(n) \leq O_{recognition} \leq O(n^2 \cdot m^3).$$

In the case of well distinguishable models, the complexity comes close to the above discussed *best case*. An example where the system slows down is shown in Figure 16, a cluttered scene which consists of three composer busts. The system detects the correct models and computes the correct locations, but due to noisy data and similar features, the discriminative power is smaller, and the overall recognition process is slower.

4.2 Recognition of 3D objects using 2D groupings

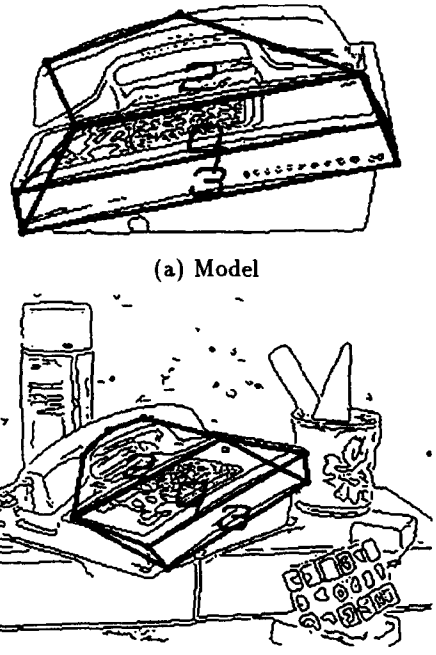
Here we are interested in the recognition of a three-dimensional object in a two-dimensional scene. This raises the question of how we describe an object and which underlying primitives we use.

Most approaches are based on low level primitives (points, line segments, and other local features). In object recognition, this strategy works only when a perfect geometric model is available. Therefore, by matching a model with a scene results in a set of correspondence hypotheses and further verification by using geometric constraints leads to the correct solution, but the computational complexity is large.

We instead propose an alternate approach using groupings. Using perceptual grouping in computer vision is an old idea, but very few implementations have been demonstrated [Lowe, 1985; Mohan and Nevatia, 1989b; Huttenlocher and Wayner, 1991; Sha'ashua and Ullman, 1988].

Here, we develop a feature hierarchy which can be used for object recognition of three-dimensional objects from a two-dimensional scene. In this hierarchy, we propose specific groupings based on proximity, parallelism, symmetry, and closure. The detection of these features is performed in an efficient way using *proximity indexing*. First, we generate features with multiple representations

to overcome unreliability of local algorithms in the preprocessing, and to handle noise and capture different levels of detail. Later, we merge perceptual similar features at higher levels of the feature hierarchy. As models, we use a set of non-registered views of a 3-D object. While most other systems use spatial correspondences to verify matching hypotheses, we use high level features and their *topological* relationships for the recognition process. These features are grouped based on closure and proximity to generate so called *high level groupings* which are stored in a table. Using indexing, we retrieve matching hypotheses, which are verified against each other with respect to topological constraints. Groups of consistent hypotheses represent detected model instances in a scene. A more detailed description can be found elsewhere in these proceedings [Stein and Medioni, 1992]. An example of successful recognition is shown in Figure 20.



(a) Model

(b) Scene

Figure 20: Recognition based on High Level Groupings

4.3 Application to the "drop-off" problem

Navigation using maps requires the frequent updating of the location of an observer with respect to the map. Humans performing this task use a large number of different problem solving techniques, both expectation driven, where a landmark is selected on the map and searched for, or data driven, where image features are extracted first and matched against the map.

The problem most often addressed by previous researchers is that of updating the current position, given a good initial solution. This can be solved by correcting (small) errors between the predicted and observed aspects of the scene.

Here instead, we study the "drop-off" problem. An excellent overview of the whole problem is given in [Thompson *et al.*, 1990]. The authors describe a preliminary computational model for the drop-off problem (the name *drop-off* comes from the case in which an observer is "dropped off" into a unfamiliar environment and has to orient himself). The observer stays at one position and tries to find his location based on visible landmarks and salient features. In our approach, the observer tries to establish his location based on the curve described by the panoramic horizon (as explained later) which is visible from his viewpoint.

It should be noted that people experience serious difficulties in solving such localization problems, leading many researchers to suggest that traditional object recognition strategies are unlikely to succeed [Thompson *et al.*, 1990]. This is based on the observation that the combinatorics of the problem are very unfavorable, as the shapes are complex and the number of different aspects is extremely large.

We challenge this view and propose that a table-based matching strategy [Stein and Medioni, 1990; Stein and Medioni, 1991] can overcome the limitations mentioned previously. We propose to extract from many locations in the map the panoramic horizon curves, which correspond to the crest line perceived by the observer as he completes a full 360° view in place. Such curves are encoded and stored in a table. To locate an unknown location, we first extract the panoramic horizon curve of the unknown location. Then we approximate it by a family of polygons with different line fitting tolerances. By using indexing into the table, we retrieve candidate locations. The correct candidate (the closest one) is found by applying further geometrical constraints during the verification step.

It should be clear to the reader that this approach is not guaranteed to provide a unique answer, as it is easy to come up with counter examples (planar or repetitive environment), but we show that it gives excellent results in complex, real environments, as shown below.

Definitions

Map: A map in our definition is a topographic map which provides us with the three dimensional coordinates for each surface point (e.g. range data). The z component (altitude or height) can be retrieved from the x and y coordinates: $z = \mathrm{Map}(x, y)$. We assume that the map is small with respect to the associated planet sphere. Therefore the map can be considered as a small planar patch. The z axis is perpendicular to this patch.

Viewer or Observer: The viewer is an observing platform with coordinates $v = (x_v, y_v, z_v)$., and the camera is at $z_v = \mathrm{Map}(x_v, y_v) + h$, where h is the height of the camera.

Horizon: The horizon is the upper bound of the projection of all landscape points on a cylinder around the observer as shown in Figure 21. Such a panoramic horizon H is a periodic curve. Projecting the curve back on the landscape results in a set of three dimensional points $\bar{H}$ whose projection on the map is shown in Figure 22. H

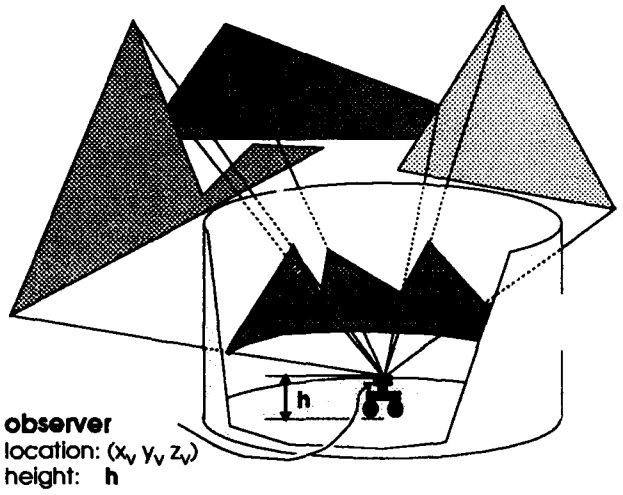


Figure 21: Horizon

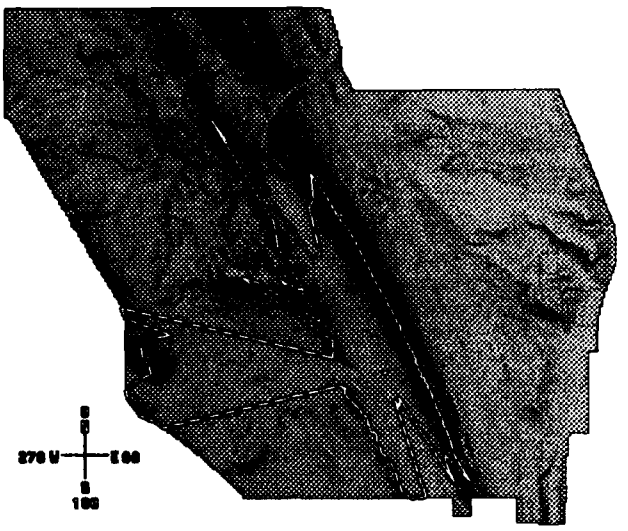


Figure 22: Map with Superimposed Reprojected Horizon

and $\bar{\mathsf{H}}$ are defined in the following way:

$$\mathsf{H}(v) = \{\frac{\Delta z(d)}{d} \mid \max_d\{\frac{\Delta z(d)}{d}\}, 0° \leq \theta < 360°\} \quad (1)$$

$$\bar{\mathsf{H}}(v) = \{p_v(\theta, d) \mid \max_d\{\frac{\Delta z(d)}{d}\}, 0° \leq \theta < 360°\} \quad (2)$$

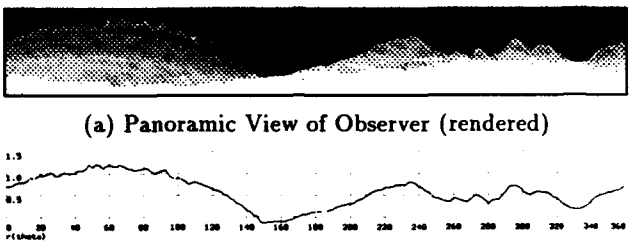
with

$$\begin{aligned} v &= (x_v, y_v, z_v) \\ p_v(\theta, d) &= (x_v^\theta, y_v^\theta, z_v^\theta) \\ d &= \sqrt{(x_v^\theta - x_v)^2 + (y_v^\theta - y_v)^2} \\ \Delta z(d) &= z_v - z_v^\theta \end{aligned}$$

We call $r(d) = \frac{\Delta z(d)}{d}$ the tangent of α, the *relative height*. Without loss of generality we set $0° = 360°$ as the North direction and define the orientation of a horizon as clockwise (as seen from above). An example can be seen in Figure 22. We display a horizon as a graph as seen from the observer. The abscissa represents the direction θ ($0° =$ north), the ordinate represents the relative height r. The graph of the horizon of Figure 22 can be seen in Figure 23.

Representation We represent the map by a set of horizons computed for the points of a grid superimposed on the map. The spacing of the grid determines the accuracy with which we can find the correct location. The horizons are stored in a table, implemented as a hash table. A table allows efficient storage (only pointers are recorded), the indexing scheme allows fast access, and different super segments with the same keys can be stored in cellar like buckets. The table (data base) grows in size with the number of recorded horizons. This process of building the data base from the map is performed off line.

Results The localization algorithm is now illustrated with an example from a real terrain map. The horizons which is seen by the observer is simulated. For the



(a) Panoramic View of Observer (rendered)

(b) Extracted Horizon Curve

Figure 23: Graph of Horizon

presentation of the range data, we always display the artificially shaded images.

As a terrain map we use the DEM (digital elevation model) covering the Martin Marietta ALV test area. A digital elevation model is a two-dimensional array of uniformly spaced terrain elevation measurements. Our DEM map consists of 810pixel×702pixel with a pixel corresponding to a size of $5 \times 5\text{m}^2$ on the ground. The whole map corresponds to an area of approximately $4 \times 3.5\text{km}^2$. Outside of the map we assume flat surface (sometimes the horizon line is not limited to the area described by the map). The horizons on the map are sampled on a grid with a grid spacing of 15 pixels. This corresponds to 75 m. For better visibility we exaggerate the elevation data. The lowest elevation is 21 (in pixels), the highest is 252. The height of the observer above the surface was always constant (2 pixels). For the linear approximation we used the line fitting tolerances 2, 3, 4, and 5. To obtain useful linear approximations, we scale the relative height by a factor of 50. These are no critical values, as significant deviations from these values do not affect the results. The localization takes less than 1 minute.

The observer is located in a valley between two ridges (see Figure 22). Figure 23 shows the rendered panoramic

view from the location of the observer (a) and the extracted horizon curve (b). After the matching the system finds the hypotheses. The best hypothesis can be seen in a closeup in Figure 24(b). Figure 25 shows the rendered view seen by the observer in comparison with the rendered panoramic view of the detected result.

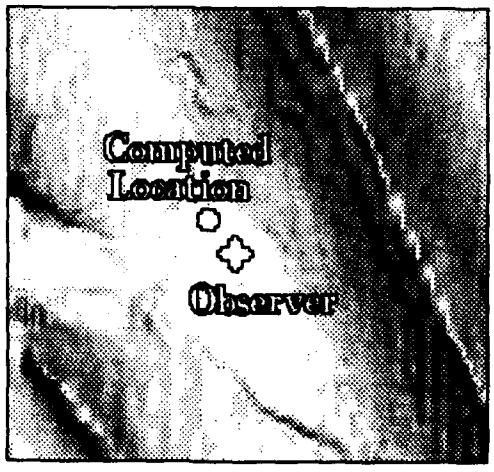
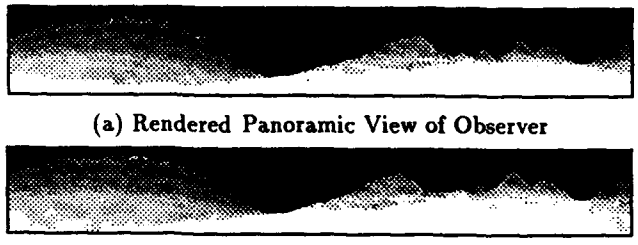


Figure 24: Closeup of the Detected Result



(a) Rendered Panoramic View of Observer

(b) Rendered Panoramic View of Computed Location

Figure 25: Simulated Panoramic Views for Horizon

5 MOTION ANALYSIS

We have had a number of projects in the analysis of sequences of images including analysis of closely spaced images, feature based analysis, motion estimation techniques, and navigation using recognition of visual features. Autonomous navigation provides the context for much of the work, though these techniques have a much broader utility.

Motion analysis using feature point analysis techniques and multiple frames forms the central focus of our work. This approach involves extracting a set of consistent features from a sequence of images, finding the corresponding features in consecutive frames, and finally computing the three-dimensional motion based on the correspondences, which also provides an estimate of the structure of the moving objects or scene. These are often described separately or as sequential operations, but integration into a single system and feedback to earlier processing is a major part of the work.

Our effort includes several separate and related projects including: analysis of closely spaced images (spatio-temporal analysis) using features such as lines, corners, and regions to extract three-dimensional structure information; matching edge based contours in a sequence of images; integrating several feature detection and matching techniques to derive three-dimensional motion and structure estimates; study of the formulation of the motion estimation problem; detection of moving objects in a scene with a moving observer; and the visual guidance of a mobile robot. Several of these projects have been completed in the past year with the resulting thesis being produced. This overview discusses the current status of the research in these areas. Some of these a are covered in more detail in other papers in this proceedings or in other recent conference papers.

5.1 Spatio-Temporal Analysis

The goal of our work in spatio-temporal analysis is to generate a dense optic flow map from a motion sequence. Because of the sparseness of 0D features (e.g. corners) or 1D features (e.g. curves), we feel that 2D features (e.g. regions) are more likely to produce dense motion estimates. Early work in spatio-temporal analysis includes that of [Bolles *et al.*, 1987]. Our work began with [Peng and Medioni, 1988; Peng and Medioni, 1989], with the extraction of paths in slices taken in the temporal direction of the spatio-temporal data volume (i.e. paths of an object point through time and space). This produces an image velocity estimate only along object contours.

In order to generate a dense displacement field, more analysis of the slice data is needed. Strips that correspond to trapezoidal regions found in the slices through the temporal dimension of the image volume are constructed for selected orientations throughout the image. These extracted strips provide estimates of the velocity component along the slice orientation. The velocity estimates of different slice orientations are combined to compute the velocity constraint for each pixel. A voting scheme is used to extract the position of the Focus of Expansion, which can then be used to compute the real velocity of the pixels.

This process is very expensive (requiring hours on serial machines), but most of the computation is easily performed on the SIMD architecture of the Connection Machine. This algorithm was transferred to a CM-2 with very good results for computational speed-up. Much of this work was reported on in previous years and has now been completed with more detail in [Peng, 1991].

5.2 Motion Estimation

We have continued our exploration of techniques for computing structure from motion using feature matches through multiple frames. The use of multiple (as opposed to two) frames is desirable for several reasons:

- to increase the robustness of the solution,
- to allow recovery of structure/motion with fewer features being tracked, and
- to allow estimation of "higher order derivatives" of the motion.



Figure 26: Reconstructed/Extrapolated Trajectories for the Rocket Field Sequence (Based on Frames 0 through 6): Data for Frames 0 through 19 Overlaid on Image 0

Pnt	x	y	z	Depth	% Error
1	11.57	-11.56	37.23	38.98	
	14.19	-10.38	40.28	42.71	9.56
3	4.41	-9.10	26.71	27.07	
	5.60	-8.43	28.09	28.64	5.80
5	-7.33	-9.88	30.72	31.59	
	-6.28	-7.87	28.53	29.21	-7.51
7	16.41	-10.49	31.46	35.48	
	18.69	-9.24	34.02	38.81	9.39
9	0.53	-9.33	27.29	27.29	
	0.87	-7.70	25.91	25.93	-5.00
11	-3.08	-8.83	66.69	66.77	
	-2.36	-5.14	97.18	97.21	45.60
13	-2.43	-11.10	35.65	35.74	
	-1.71	-8.74	33.36	33.40	-6.53
15	0.82	-11.68	44.19	44.20	
	1.71	-5.88	37.30	37.34	-15.51
17	21.32	-15.61	52.63	56.79	
	23.96	-13.22	58.36	63.08	11.09

Table 1: Reconstructed Structure Compared to Ground Truth Values for the Rocket Field Sequence

We have completed development and implementation of an algorithm for the shape from motion problem given point feature correspondences and perspective projection. This solution works for a class of motions called chronogeneous motion, which includes uniform acceleration and constant angular velocity rotation and translation as special cases. The solution is by an iterative algorithm that recovers the three-dimensional motion of the feature points and the three-dimensional location of each feature in each frame. An additional closed form algorithm that recovers motion and structure for uniform acceleration is used to generate initial guesses for the iterative procedure [Franzen, 1991a].

These algorithms are discussed further in these proceedings [Franzen, 1992] with additional results, or in the thesis [Franzen, 1991b]. Figure 26 shows an example of reconstructed trajectories of objects in the image plane based on the computed three-dimensional motion. Figure 1 illustrates the structure accuracy (compared to measured ground truth) for this problem. These results show that this algorithm performs well in recovering structure and motion parameters from feature point correspondences. We are using this motion estimation technique in our other motion work [Kim and Price, 1992].

5.3 Integrated System for Motion

We have continued to develop and use an integrated system for testing and combining each of the subsystems of the motion analysis system (segmentation, feature extraction and matching, motion estimation, motion feedback to matching). In a paper in these proceedings [Kim and Price, 1992], we present an approach to improve the results of matching features using a feature-based motion analysis technique applied to multiple frames. Automatic correspondence procedures produce noisy results due to variations in the feature extraction process, errors in the matching process, or noise in the actual motion. Tracking the same feature through a number of frames results in fragmented trajectories due to occlusions or missing matches.

We developed a technique that gradually refines the initial noisy correspondence data by using the future trajectory of matched points based on the estimated motion parameters and eliminating those points that are not compatible with the others in the sequence and with the estimated motion. We also links fragments of a single feature into a single trajectory using the three-dimensional motion estimation to indicate which image points should correspond to the same real feature.

Results from this system are illustrated in figure 27 which shows the initial correspondence data used as input to the system and the refined and linked correspondence data used for the final motion estimation. The trajectories are drawn on the first image of the sequence and correspond to sequences that may start and end at any frame in the fifteen frame sequence.

5.4 Mobile Platform

We have used our mobile robot as a testbed for studying simple processing techniques that can be used for visual guidance. On this platform we have implemented a simple sonar-based object avoidance procedure that will prevent collisions with objects and allows the vision system to concentrate on other navigation goals. We are developing a trinocular (three camera) stereo system for reliable generation of basic three-dimensional descriptions of the scene for navigation.

6 AERIAL IMAGE ANALYSIS

Our work in aerial image analysis consists of two major components. First is the transfer of technology funded



(a) Initial correspondence



(b) Final correspondence results

Figure 27: Correspondence data used in motion estimation

by DARPA to the RADIUS program. Specifically, we are focussing on the transfer of our techniques for building detection in aerial images [Huertas and Nevatia, 1988b; Mohan and Nevatia, 1989c]. Second project is the continuation of our long range effort of analyzing complex cultural domains. We have chosen large commercial airports as a test domain. In previous work [Huertas *et al.*, 1990b; Huertas *et al.*, 1989b] have shown good results on the detection of runways and taxiways. In this report, we describe our recent work on aircraft detection.

6.1 Aircraft Detection

We are building a program to detect commercial and private aircraft in aerial images. There are many types of aircraft that can be present on a commercial airport complex including Boeing 737 and 747, Lockheed 1011, and Dehavilland DHC-6. These aircraft are varied in shape and size. For instance, some have engines on the rear fuselage, some on the wings and some are front propelled. We would like to avoid having a different detailed model for each aircraft in our system, as is the case in [Heller and Mundy, 1990] for example. Most aircraft in the commerical airport setting have a fuselage and two wings placed symmetrically relative to the fuselage. We will use this basic model to hypothesize aircraft in a two step process.

The first step of this process assumes a reasonable contrast between the aircraft and the surface on which the aircraft sits. The example we use is a 2048x2048 portion of Boston Logan Airport 28. First step of this process is to detect linear segments 29. Next, we find anti-parallels which are possible candidates for aircraft fuselages 30. These anti-parallels can correspond to the fuselage section in front of the wings (front fuselage) or the fuselage section behind the wings (rear fuselage). In order to instantiate a hypothesis for an aircraft, the fuselage sections must be supported by a wing pair. In this step, the wings are sets of line segments placed symmetrically relative to the candidate fuselage within certain angular and distance constraints. The angle of the wing relative to the fuselage varies according to the aircraft type, however the variance is reasonable to constrain the search. If a fuselage has more than one wing- pair match, we group the fuselage hypothesis anti-parallels using the co-linear grouping functions from the transportation network system to try to establish one solid fuselage hypothesis. After fuselage fragment grouping, we seek further support for the aircraft hypothesis by looking for further image evidence of wings being interrupted by aircraft engines. Further, engine placement can be a cue to aircraft type identification.

However, often there is very little contrast between an aircraft and the runway/terminal area on which it sits. In this case, we rely on shadow information in the second step of our aircraft identification process. In this case, we detect candidate fuselage sections by looking for thin dark anti-parallels 31 corresponding to aircraft shadow hypotheses. The wing pair detection process is also the same, except we concentrate on finding dark anti-parallels corresponding to the wing shadows. We do not go back into the data and look for further evidence because it is assumed that there is insufficient contrast to find any more details.

Our final set of hypotheses correspond to a conjunction of the aircraft detected from the ribbon description 32 and those found using the shadow evidence. We hope to improve this module by integrating the two steps to give higher confidence to aircraft detected explicitly from their general shape which also have shadow evidence. We plan to use the aircraft hypothesis in our system to help constrain the search for complex building structures, and to improve our airport transportation network detection system.

7 PARALLEL PROCESSING

The parallel processing group at USC, in collaboration with the vision group has been actively involved in devising efficient parallel solutions to problems from all levels of image understanding [Prasanna Kumar, 1991]. Our research is focused on understanding the key problems in parallelizing the techniques developed by the vision community (in particular those developed by the vision group at USC), and also implementing the solutions on available parallel machines. A summary of our current work is outlined below.

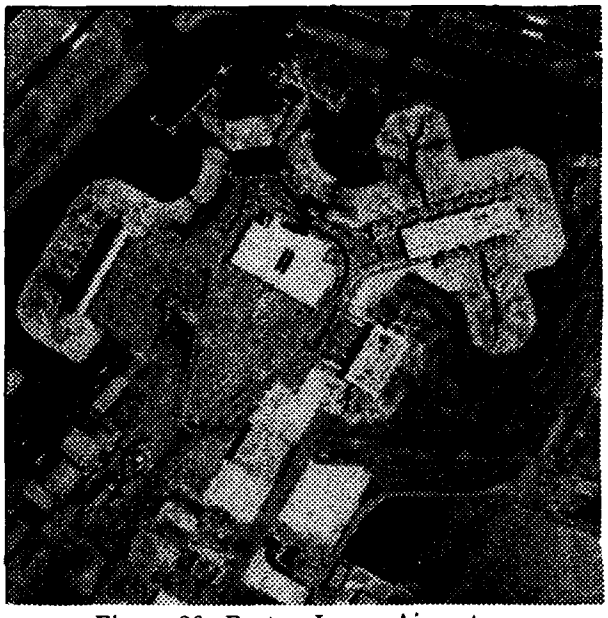


Figure 28: Boston Logan Airport

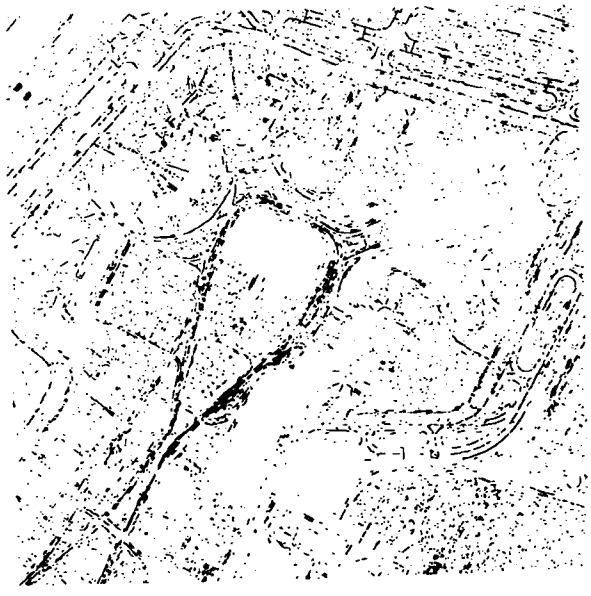


Figure 30: Fuselage Anti-parallels

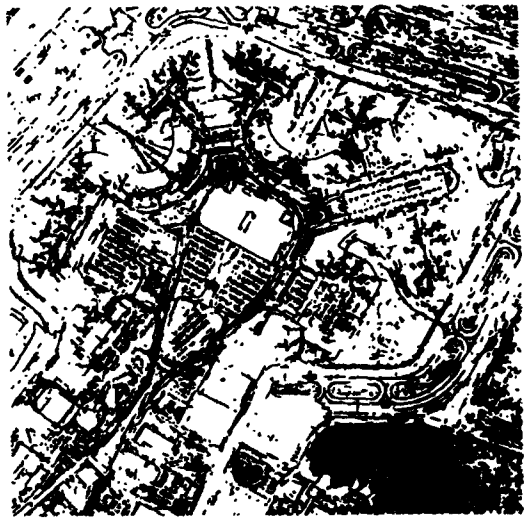


Figure 29: Linear Segments



Figure 31: Shadow Anti-parallels

Stereo and Image Matching Stereo matching is one of the well known methods for extraction of depth information. Depth recovery is a crucial problem in image understanding with applications in robotics and navigation. For stereo matching, we have proposed $O(\frac{Nn^3}{P})$ time algorithm on a P processor fixed size linear array, where N is the number of line segments in one image, n is the number of line segments in a window determined by the object size, and $P \leq n$ [Khokhar *et al.*, 1991]. This algorithm is a parallel implementation of the stereo matching algorithm proposed by Medioni and Nevatia in [Medioni and Nevatia, 1985].

Discrete relaxation techniques have been widely used in computer vision and artificial intelligence. For the image matching problem, discrete relaxation technique outlined in [Medioni and Nevatia, 1984] leads to a sequential execution time of $O(n^3m^3)$ for labelling n objects with m labels. In [Lin and Prasanna, 1991] we have proposed a faster sequential algorithm for image matching which runs in $O(n^2m^2)$ time, where n is the number of line segments in the image and m is the number of line segments in the model. Also, a partitioned parallel implementation has been developed by using the proposed sequential algorithm. $O((\frac{nm}{P}+P)nm)$ time performance is achieved on a P processor fixed size linear array, where $P \leq nm$.



Figure 32: Aircraft Hypotheses

Sorting on Reconfigurable Mesh The Reconfigurable Mesh forms the CAAPP level of Image Understanding Architecture (IUA) [Weems *et al.*, 1989]. An optimal sorting algorithm on the Reconfigurable Mesh is derived in [Jang and Prasanna, 1991]. The algorithm sorts n numbers in constant time using $n \times n$ processors. The best known previous result uses $O(n \times n \log^2 n)$ processors. Our algorithm satisfies the AT^2 lower bound of $\Omega(n^2)$ for sorting n numbers in the word model of VLSI. Modification to the algorithm for area-time tradeoff is shown, to achieve the AT^2 lower bound over $1 \leq T \leq \sqrt{n}$. Previously, the lower bound was achieved over $\log n \leq T \leq \sqrt{n}$. Notice that, using sort as a basic procedure number of low- and intermediate-level Image Understanding problems can be solved on the IUA.

Graph Algorithms Many of the intermediate-level computer vision tasks can be posed as graph problems. Particularly, digitized picture graphs (DPGs) of two and three dimensions are of primary importance due to their natural correspondence with black/white images. We introduce a notion of *partitionability* of graphs and show that DPGs (of any fixed dimension) are partitionable [Rao and Prasanna, 1991]. This partitionability property helps in constructing efficient parallel algorithms for many problems on digitized picture graphs. We show that our techniques can be efficiently simulated on a $P \times P$, fixed-size mesh-connected computer, $1 \leq P \leq n$. Unlike other approaches [Miller and Stout, 1985], our algorithms, because of the partitionability idea, easily extend to problems in higher dimensions.

VLSI Architectures for Image Transforms and Vector Quantization We have studied VLSI architectures for various image transforms and vector quantization techniques. Two linear array architectures have been proposed for computing the arithmetic Fourier transform and image compression using vector quantization [Park and Prasanna, 1991a; Park and Prasanna, 1991b]. These architectures have modular PEs and can support real-time processing. The designs can operate with less number of PEs than the input size. The proposed designs require fixed I/O bandwidth with the host.

Parallelization of Symbolic Techniques in Vision There is relatively little work done in parallelizing high level vision algorithms. Such algorithms are usually symbolic in nature and the processing is not entirely local. We believe that when dealing with such complex algorithms, the parallel implementation must be concerned with the following four characteristics: algorithm speedup, processor efficiency, system complexity, and programmer burden.

Most research in parallel processing has been concerned solely with the first two characteristics. We have been pursuing an alternative that achieves a better balance between the desired characteristics. In our approach we classify algorithms, in terms of operations, data dependencies, data movements, and algorithm characteristics, and then specify a parallel processor architecture that is well suited to those characteristics [Reinhart, 1991].

We have applied this methodology to a number of mid and high level vision algorithms. Our first experience was with an algorithm for image matching via relaxation labelling with symbolic objects and geometric constraints[Medioni and Nevatia, 1984]. Our analysis indicated that the use of an MIMD architecture that comprises powerful processing elements programmed with the loosely synchronous protocol. A suitable interconnect topology is one of logarithmic diameter. Two implementations were developed, one using binary tree and the other using hypercube connections. This scheme exploits the coarse grain parallelism within the algorithm. Further analysis shows that equipping each PE with a tightly coupled vector processor would exploit the fine grain parallelism within the algorithm. This architecture achieves high degrees of speedup and efficiency while using software that is nearly identical to that of the serial implementation, thus system complexity and programmer burden are minimized. Details of this study were previously reported in [Reinhart and Nevatia, 1990].

In more recent work, we have studied an algorithm for object recognition that uses graph matching. Our analysis again indicates that an MIMD architecture with powerful processing elements is suited to this problem. The PEs are connected by a hypercube topology. Again, we achieve significant algorithm speedup and processor efficiency while using software that is nearly identical to that of the serial implementation.

Lastly, we have studied the mid-level operations of linear feature extraction and perceptual organization. These operations may appear to be simple and repetitive, and thus well suited to SIMD implementations. However, this is not the case. Typically, parallel implementations only focus on the study of a specific algorithm. They assume that the input is given in the desired form and the output is produced in some form. In a system (or sub-system) that comprises of a number of processing steps, conversion of the output of one stage

to another itself can be a majo step, possibly requiring serial implementation. A simple example is that of linear feature extraction where finding edges and then their neighbors that would form curves is an iconic process that is easily implemented on a SIMD machine. However, this is different from actually producing a *list* of curves, each curve given by a list of points forming it, in order, and possibly a linear approximation to it as well. This is the structure needed for subsequent use of the linear feature processing. Our proposed implementation is described in detain in another paper in these proceedings [Reinhart and Nevatia, 1992].

In conclusion, our research has shown that the complex operations and data movements required by mid and high level vision can be performed efficiently if care is taken in specifying the parallel processor architecture. Implementations that achieve high degrees of algorithm speedup and processor efficiency can be attained without sacrificing system complexity and programmer burden. Such architectures can be realized utilizing heterogeneous designs or via reconfigurable architectures given an efficient reconfiguration procedure. We have certainly not studied all the algorithms used in vision, but believe that our choice of selected algorithms covers enough of a span to indicate that it is fruitful to further pursue this approach. We also believe that the next step is to investigate the parallel implementation of complete, heterogeneous vision systems that comprise low, mid, and high-level algorithms.

8 Acknowledgments

This paper represents the work of many different current and former students, visitors and staff members. J. Y. Cartoux and D. Kim have worked on the mobile robot project. Y. Chen is working on descriptions from multiple views of objects., R. Chung is working on shape inference from stereo data. S. Cochran finished a thesis on stereo analysis. A. Huertas and W. Cole are working on the aerial image analysis project., W. Franzen finished a thesis on motion and structure estimation., G. Guy is working on perceptual grouping techniques. Y. C. Kim has worked on the integrated motion system., C. Liao is working on deformable models., B. Parvin has worked on generating boundary representations from range data., S. L. Peng finished a thesis on spatio-temporal analysis., H. Rom worked on shape decompositions., F. Stein worked on 3-D object recognition and the horizon recognition problem, F. Ulupinar finished a thesis on 3-D descriptions from contour data., M. Zerroug has begun new work on descriptions from incomplete contours..

References

[Bolles *et al.*, 1987] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision*, 1:7–55, 1987.

[Chen and Medioni, 1991] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Proceedings of the IEEE Conference on Robotics and Automation*, volume 3, pages 2724–2729, April 1991.

[Chung and Nevatia, 1991] C.-K. R. Chung and R. Nevatia. Use of monocular groupings and occlusion analysis in a hierarchical stereo system. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 50–56, Maui, Hawaii, June 1991.

[Chung and Nevatia, 1992] C.-K. R. Chung and R. Nevatia. Recovering LSHGCs and SHGCs from stereo. In *Proceedings of the DARPA Image Understanding Workshop*, San Diego, CA, January 1992. In these Proceedings.

[Cohen *et al.*, 1991] I. Cohen, L. Cohen, and N. Ayache. Introducing new deformable surfaces t gment 3D scenes. In *Proceedings of the Conference ... Computer Vision and Pattern Recognition*, pages 738–739, Maui, HI, June 1991.

[Fan *et al.*, 1989] T.-J. Fan, G. Medioni, and R. Nevatia. Recognizing 3-d objects using surface descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1140–1157, November 1989.

[Franzen, 1991a] W. Franzen. Structure and motion from uniform 3D acceleration. In *Proceedings of the Workshop on Visual Motion*, pages 14–20. IEEE Computer Society, Oct 7–9, 1991.

[Franzen, 1991b] W. Franzen. *Structure from Chronogeneous Motion.* PhD thesis, University of Southern California, May 1991. IRIS Technical Report 266.

[Franzen, 1992] W. O. Franzen. Structure from chronogeneous motion: A summary. In *Proceedings of the DARPA Image Understanding Workshop*, San Diego, CA, January 1992.

[Heller and Mundy, 1990] A.J. Heller and J.L. Mundy. Benchmark evaluation of a model-based object recognition system. In *Proceedings of the DARPA Image Understanding Workshop*, pages 727–741, Boston, Massachusetts, May 1990.

[Huertas and Nevatia, 1988a] A. Huertas and R. Nevatia. Detecting buildings in aerial images. *Computer Vision, Graphics, and Image Processing*, 41(2):131–152, February 1988.

[Huertas and Nevatia, 1988b] A. Huertas and R. Nevatia. Detecting buildings in aerial images. *Computer Vision, Graphics, and Image Processing*, 41(2):131–152, February 1988.

[Huertas *et al.*, 1989a] A. Huertas, W. Cole, and R. Nevatia. Using generic knowledge in analysis of aerial scenes: A case study. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1642–1648, Detroit, MI, August 1989.

[Huertas *et al.*, 1989b] A. Huertas, W. Cole, and R. Nevatia. Using generic knowledge in analysis of aerial scenes: A case study. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1642–1648, Detroit, Michigan, August 1989.

[Huertas *et al.*, 1990a] A. Huertas, W. Cole, and R. Nevatia. Detecting runways in complex airport

scenes. *Computer Vision, Graphics, and Image Processing*, 51(2):107–145, August 1990.

[Huertas *et al.*, 1990b] A. Huertas, W. Cole, and R. Nevatia. Detecting runways in complex airport scenes. *Computer Vision, Graphics, and Image Processing*, August 1990.

[Huttenlocher and Wayner, 1991] D. P. Huttenlocher and P. C. Wayner. Finding Convex Edge Groupings in an Image. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 406–412, Maui, Hawaii, June 1991.

[Jang and Prasanna, 1991] J. Jang and V. K. Prasanna. An optimal sorting algorithm on reconfigurable mesh. Technical report, University of Southern California, Los Angeles, 1991.

[Kanizsa, 1976] G. K. Kanizsa. Subjective contours. *Scientific American*, April, 1976.

[Kass *et al.*, 1987] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 259–268, London, England, June 1987.

[Khokhar and Prasanna, 1992] A. Khokhar and V. K. Prasanna. Parallel algorithms for stereo and image matching. In *Proceedings of the DARPA Image Understanding Workshop*, San Diego, CA, January 1992. In these Proceedings.

[Khokhar *et al.*, 1991] A. Khokhar, W. Lin, and V. K. Prasanna. Image matching and stereo matching on fixed size linear arrays. In *Computer Architectures for Machine Perception Workshop*, 1991.

[Kim and Price, 1992] Y. C. Kim and K. Price. Improved correspondence in multiple frame motion analysis. In *Proceedings of the DARPA Image Understanding Workshop*, San Diego, CA, January 1992.

[Lim and Binford, 1988] H. S. Lim and T. O. Binford. Curved surface reconstruction using stereo correspondence. In *Proceedings of the DARPA Image Understanding Workshop*, pages 809–819, Cambridge, Massachusetts, April 1988.

[Lin and Prasanna, 1991] W. Lin and V. K. Prasanna. Parallel algorithms and architectures for discrete relaxation technique. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 1991.

[Lowe, 1985] D. G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Hingham, MA, 1985.

[Lowe, 1987] D. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[Medioni and Nevatia, 1984] G. Medioni and R. Nevatia. Matching images using linear features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):675–685, November 1984.

[Medioni and Nevatia, 1985] G. Medioni and R. Nevatia. Segment-based stereo matching. *Computer Graphics and Image Processing*, 31(1):2–18, July 1985.

[Menet *et al.*, 1990] S. Menet, P. Saint-Marc, and G. Medioni. B-snakes: Implementation and application to stereo. In *Proceedings of the DARPA Image Understanding Workshop*, pages 720–726, Pittsburgh, Pennsylvania, September 1990.

[Miller and Stout, 1985] R. Miller and Q. F. Stout. Geometric algorithms for digitized pictures on a mesh-connected computer. In *IEEE Transactions on Patt. Anal. and Mach. Intel. (PAMI)*, 1985.

[Mohan and Nevatia, 1989a] R. Mohan and R. Nevatia. Segmentation and description based on perceptual organization. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 333–341, San Diego, California, June 1989.

[Mohan and Nevatia, 1989b] R. Mohan and R. Nevatia. Using perceptual organization to extract 3-d structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1121–1139, November 1989.

[Mohan and Nevatia, 1989c] R. Mohan and R. Nevatia. Using perceptual organization to extract 3-D structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1989.

[Park and Prasanna, 1991a] H. Park and V. K. Prasanna. VLSI architectures for computing the arithmetic fourier transform. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 1991.

[Park and Prasanna, 1991b] H. Park and V. K. Prasanna. VLSI architectures for real time vector quantization. In *International Symposium on Circuits and Systems*, 1991.

[Parvin and Medioni, 1991a] B. Parvin and G. Medioni. A dynamic system for object description and correspondence. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 393–399, Maui, June 1991.

[Parvin and Medioni, 1991b] B. Parvin and G. Medioni. A layered network for the correspondence of 3d objects. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1808–1813, April 1991.

[Parvin, 1991] B. Parvin. *A Dynamic System for Object Description and Correspondence*. PhD thesis, University of Southern California, 1991.

[Peng and Medioni, 1988] S.-L. Peng and G. Medioni. Spatio-temporal analysis of velocity estimation of contours in an image sequence with occlusion. In *Proceedings of the International Conference on Pattern Recognition*, pages 236–241, Rome, Italy, November 1988.

[Peng and Medioni, 1989] S.-L. Peng and G. Medioni. Interpretation of image sequences by spatio-temporal analysis. In *Proceedings of the Workshop on Visual Motion*, pages 236–241, Irvine, California, March 1989.

[Peng, 1991] S.-L. Peng. *Image Sequence Analysis Using Temporal Slices.* PhD thesis, University of Southern California, January 1991. IRIS Technical Report 265.

[Pentland and Sclaroff, 1991]
A. P. Pentland and S. Sclaroff. Closed-form solutions for physically based shape modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):715–729, July 1991.

[Prasanna Kumar, 1991] V. K. Prasanna Kumar. *Parallel Algorithms and Architectures for Image Understanding.* Academic Press, Boston, 1991.

[Rao and Prasanna, 1991] A. S. Rao and V. K. Prasanna. Efficient parallel algorithms for k-dimensional images. In *International Conference on Image Analysis and Processing*, 1991.

[Reinhart and Nevatia, 1990] C. Reinhart and R. Nevatia. Efficient parallel processing in high level vision. In *Proceedings of the DARPA Image Understanding Workshop*, Pittsburgh, PA, September 1990.

[Reinhart and Nevatia, 1992] C. Reinhart and R. Nevatia. Parallel linear feature extraction. In *Proceedings of the DARPA Image Understanding Workshop*, San Diego, CA, January 1992.

[Reinhart, 1991] C. Reinhart. *Specifying Parallel Processor Architectures for High-Level Computer Vision Algorithms.* PhD thesis, University of Southern California, 1991.

[Requicha, 1980] A. A. G. Requicha. Representations for rigid solids: Theory, Methods, and Systems. *ACM Computing Surveys*, 12(4):437–464, 1980.

[Rom and Medioni, 1992] H. Rom and G. Medioni. Hierarchical decomposition and axial representation of shape. In *Proceedings of the DARPA Image Understanding Workshop*, San-Diego, California, January 1992. (these Proceedings).

[Saint-Marc and Medioni, 1990]
P. Saint-Marc and G. Medioni. B-spline contour representation and symmetry detection. In *First European Conference on Computer Vision*, pages 604–606, Antibes, France, April 1990.

[Sha'ashua and Ullman, 1988] A. Sha'ashua and S. Ullman. Structural saliency: the detection of globally salient structures using a locally connected network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 321–327, Tampa, FL, 1988.

[Stein and Medioni, 1990] F. Stein and G. Medioni. Efficient Two Dimensional Object Recognition. In *Proceedings of International Conference on Pattern Recognition*, Atlantic City, New Jersey, June 1990.

[Stein and Medioni, 1991] F. Stein and G. Medioni. Structural Hashing: Efficient Three Dimensional Object Recognition. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, Maui, Hawaii, June 1991.

[Stein and Medioni, 1992] F. Stein and G. Medioni. Recognition of 3-D Objects from 2-D Groupings. *Proceedings of the DARPA Image Understanding Workshop*, January 1992. (These Proceedings).

[Terzopoulos and Metaxas, 1991] D. Terzopoulos and D. Metaxas. Dynamic 3d models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, July 1991.

[Terzopoulos *et al.*, 1987] D. Terzopoulos, A. Witkin, and M. Kass. Energy constraints on deformable models: Recovering shape and non-rigid motion. In *Proceedings of the National Conference on Artificial Intelligence*, pages 755–760. Morgan Kaufmann Publishers, Inc., July 13–17, 1987.

[Terzopoulos, 1988] D. Terzopoulos. The computation of visible-surface representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):417–438, July 1988.

[Thompson *et al.*, 1990] W. B. Thompson, H. L. Pick, B. H. Bennet, M. R. Heinrichs, S. L. Savitt, and K. Smith. Map-based localization: The drop-off problem. In *Proceedings of the DARPA Image Understanding Workshop*, 1990.

[Ulupınar and Nevatia, 1988] F. Ulupınar and R. Nevatia. Using symmetries for analysis of shape from contour. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 414–427, Tampa, Florida, December 1988.

[Ulupınar and Nevatia, 1990a]
F. Ulupınar and R. Nevatia. Inferring shape from contour for curved surfaces. In *Proceedings of the International Conference on Pattern Recognition*, volume 1, pages 147–154, Atlantic City, New Jersey, June 1990.

[Ulupınar and Nevatia, 1990b] F. Ulupınar and R. Nevatia. Shape from contour: SHGCs. In *Proceedings of IEEE International Conference on Computer Vision*, pages 582–582, Osaka, Japan, 1990.

[Ulupınar and Nevatia, 1991] F. Ulupınar and R. Nevatia. Recovering shape from contour for constant cross section generalized cylinders. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 674–676, Maui, Hawaii, 1991.

[Ulupınar and Nevatia, 1992] F. Ulupınar and R. Nevatia. Recovery of 3-D objects with multiple curved surfaces from 2-D contours. In *Proceedings of the DARPA Image Understanding Workshop*, San Diego, CA, January 1992.

[Weems *et al.*, 1989] C. Weems, S. Levitan, A. Hanson, and E. Riseman. The image understanding architecture. *International Journal of Computer Vision*, 2(3):251–282, January 1989.

Image Understanding Research at SRI International

Martin A. Fischler and Robert C. Bolles

Artificial Intelligence Center
SRI International
333 Ravenswood Ave., Menlo Park, CA 94025
(fischler@ai.sri.com bolles@ai.sri.com)

Abstract

The image understanding program at SRI International is a broad effort spanning the entire range of machine vision research. In this report we describe our progress in two domains: the first is concerned with modeling the earth's surface from aerial imaging sensors; the second is concerned with ground-level vision and vision-based land navigation. In particular, we describe progress in stereo compilation and automated terrain modeling from aerial imagery; in interactive scene modeling and scene generation; in automatic image segmentation and delineation of man-made objects; in detecting and tracking moving objects; and in using knowledge beyond shape and immediate appearance to recognize objects in natural scenes and other complex domains.

1 Introduction

The overall goal of Image Understanding research at SRI International is to obtain solutions to fundamental problems in computer vision that are essential in allowing machines to model, manipulate, and understand their environment from sensor-acquired data and stored knowledge.

In this report we describe progress in two domains, aerial and ground-based vision.[1] The first is concerned with modeling the earth's surface from photographs taken from aircraft and satellites; the second is concerned with modeling a natural environment in real time from data taken by a robotic device moving through, and interacting with, this environment.

In the discussion of the first domain we describe our progress in developing stereo techniques for building terrain models from aerial imagery; interactive techniques for building three-dimensional models of man-made and cultural objects, and a new automatic technique for segmenting aerial images into coherent regions and for detecting and delineating man-made objects.

In the discussion of ground-based vision we describe progress in developing techniques for building object descriptions that evolve gradually over time as more data are obtained, motion analysis techniques for detecting and tracking moving objects in data taken by moving sensors, and a new method for using contextual information to recognize natural objects, such as trees and bushes, in outdoor scenes.

An important theme in much of our current work is an emphasis on computational performance — especially through the development of algorithms capable of exploiting the new parallel machine architectures now available (e.g., the Connection Machine[tm]).[2]

2 Stereo Research and Stereo Based Modeling

Stereo reconstruction is a critical task in machine vision, with applications to robotics and cartography, that has received a great deal of attention in the image understanding community ([Barnard90], [Barnard&Fischler90]). Its importance goes beyond the obvious application to constructing geometric models: understanding scene geometry is necessary for effective feature extraction and other scene analysis tasks. While considerable success has been achieved in important parts of the problem, there is no complete stereo-modeling system that can perform reliably in a wide variety of scene domains.

Historically, the computational modeling of stereo vision has been driven by a number of diverse motivations. The practical applications of automated stereo are so important, especially in cartography and robotics, that many engineering-oriented approaches have been tried. These often use "correlation" techniques: patches of intensities in one image are searched for in the other image by maximizing a measure of correlation or minimizing a measure of intensity mismatch. The other motivations, such as the desire to model biological stereo, involve a variety of techniques. Some are feature-based: discrete

[1] Supported by various Defense Advanced Research Projects Agency contracts.

[2] Use of a Connection Machine was provided by DARPA.

local features (usually edges) are matched across images; others use an approach in which a dense disparity map is the state variable of a system, stereo matching is then formulated as an optimization problem: find the best disparity map by maximizing an objective function that measures the "quality" of the map.

Our research strategy in this task is to develop new techniques for the key steps in the stereo process, such as matching and interpolation, and, in parallel, to integrate these new ideas with existing techniques in the context of an operational system. As part of this process SRI has implemented [Hannah85] and evaluated [Hannah88, Hannah89] a complete high-performance stereo system, STEREOSYS, that uses a combination of the correlation and feature-matching approaches. In a test of existing stereo systems on 12 pairs of digital images, conducted by the International Society of Photogrammetry, STEREOSYS was able to successfully process more of the images than any other system (11 out of the 12 pairs); while no formal ranking of the test results will be published, it appears that this system placed first (or very near the top) in the competition. While STEREOSYS was originally targeted for application to the domain of aerial imagery, it has also performed successfully in extensive tests involving ground-level outdoor imagery obtained from a camera mounted on a moving platform.

Another system we have developed for stereo matching is CYCLOPS [Barnard90]. This work began as a stochastic-optimization approach to stereo matching, but has evolved into a more complete system for cartographic terrain modeling, including software modules for camera modeling, epipolar resampling, the generation of regular-grid elevation maps, ortho-images, contour plots, and synthetic perspective views, in addition to the central task of image matching. One of the goals of this work has been to develop efficient stereo-processing methods for massively parallel SIMD architectures. The CYCLOPS system is implemented on the Connection Machine. The current implementation is capable of producing a dense terrain model (depth for every pixel) for a typical pair of 1024x1024 aerial stereo image in about eight minutes, using a Connection Machine with 4096 processors. First, camera model information is used to produce corrected images with only horizontal parallax. The corrected images are then matched with a multigrid optimization algorithm. Essentially, the matching algorithm is a stochastic regularization method that tries to find the flattest dense disparity map that matches the photometry with least error. It does so by iterating a microcanonical version of simulated annealing across several levels of a resolution pyramid, using the results from the coarser levels to initialize the optimization search at the finer levels. After the corrected images are matched the disparity measurements are converted into a dense but irregular mesh of depth measurements, which is then resampled into a grid of elevations with respect to regularly spaced ground coordinates.

In some of our most recent work, we have integrated information produced by shading and stereo in the context of a shape-from-shading algorithm that can use stereo depth maps to provide initial and boundary conditions [Leclerc&Bobick91, *these proceedings*]. We note that shading and stereo are complementary techniques for two reasons: First, relative depth discrimination from stereo decreases with absolute depth, whereas orientation discrimination, as determined by shading, does not. Thus, for example, the geometric details of tree canopies in aerial images may be lost to stereo, but can be recovered from the shading information. Second, regions in the image where stereo fails because of lack of interesting visual events, such as the rolling hills of desert areas, are good candidate regions for shading analysis. Our implemented system was applied to a number of synthetic and real images. The algorithm was able to recover surface scratches and dents as little as 1mm in depth at a distance of 2m, which is well below the competence of stereo analysis for the given image resolution. We are currently attempting to use confidence measures provided by the stereo system to control the application of the shading analysis and to adjust the balance between the shading and stereo solutions.

Given a set of relatively dense depth maps of a given area (as might be provided from different views by stereo, shading, or a laser range-finder), one must still integrate the information into a consistent 3-D representation of the imaged surfaces. Fua and Sander [*these proceedings*] describe an integration approach based on fitting local quadric surfaces to the data. Their algorithm consists of four sequential steps: smoothing of the points by iterative local surface fitting; resampling the smoothed points onto a regular grid; computation of an adjacency graph of the points with clustering of the connected components; and triangulation of the clusters. This results in a representation of the depth data in terms of locally consistent quadric surfaces. The algorithm was applied to a number of complex 3-D scenes, integrating as many as 12 depth maps into a single global representation, with excellent results.

While the stereo problem remains a key focus of our research program, additional effort is now being devoted to developing an understanding of how knowledge of scene depth can be effectively used in the scene-partitioning and object-recognition tasks.

3 Interactive Techniques for Scene Modeling: A Cartographic Modeling Environment

Manual photointerpretation is a difficult and time-consuming step in the compilation of cartographic information. However, fully automated techniques for this purpose are currently incapable of matching the human's ability to employ background knowledge, common sense, and reasoning in the image-interpretation task. Near-term solutions to computer-based cartography must include both interactive extraction techniques and new ways of using computer technology to provide the end-user with useful information in the form of both image and map-like interactive computer displays.

In order to support research in semiautomated and automated computer-based cartography, we have developed the SRI Cartographic Modeling Environment (CME). In the context of an interactive workstation-based system, the user can manipulate multiple images; camera models; digital terrain elevation data; point, line, and area cartographic features; and a wide assortment of three-dimensional objects. Interactive capabilities include free-hand feature entry, feature editing in the context of task-based constraints, and adjustment of the scene viewpoint. Synthetic views of a scene from arbitrary viewpoints may be constructed using terrain and feature models in combination with texture maps acquired from aerial imagery. This ability to provide an end-user with an interactively controlled scene-viewing capability could eliminate the need to produce hard-copy maps in many application contexts. Additional applications include high-resolution cartographic compilation, direct utilization of cartographic products in digital form, and generation of mission-planning and training scenarios.

Recent work has focused on porting the CME to a UNIX/C platform (from its current LISP-machine implementation) in order to support technology transfer goals. In particular, the CME is being reimplemented on a Sun Microsystems SparcStation II using X Windows. The software base is a combination of Common Lisp with CLOS and C, where C is used to increase performance of selected time-critical components. In addition, an interface is being developed to allow C application programmers to access the CME data structures and functions that are implemented in LISP. The intent is to provide a single environment that provides the full range of CME functionality to both LISP and C programmers. Other work involves developing more flexible object representations, irregular terrain grids, and improved interfaces to other systems such as the SRI-developed Core Knowledge System. One especially important technical improvement involves sensor geometry extensions.

The SRI Cartographic Modeling Environment uses sensor geometry models in two principal ways: 1) projecting the 3D world coordinates into 2D sensor (pixel) coordinates, and 2) computing the intersection of a 3D ray (corresponding to a sensor pixel) with a terrain model. The basic CME system currently supports only central (perspective) projection and orthographic projection. In central projection, each point in 3-space is projected onto the camera sensor plane along a ray passing through a common point, the projection center. We are currently implementing a generic capability for dealing with non-central-projection sensor geometries. When accomplished, essential operations now supported for central projection imagery would also be supported for other types of (orbiting) sensors. These operations include:

1. Display of three-dimensional feature models that are cartographically registered to non-central-projection imagery.

2. Terrain rendering, using data acquired with any sensor geometry, to a format simulating any other sensor geometry. An example would be mapping non-central-projection imagery onto a terrain model and generating a simulated image showing the result viewed with a central-projection sensor.

Our earlier work in this overall task area was presented in two papers, one describing basic design issues for this type of system [Hanson,Pentland,&Quam87], and the other providing an overview of our original plans for the implementation [Hanson&Quam88]. A current description appears as a paper in these proceedings [Mundy, et. al.].

4 Detection and Delineation of Objects in Aerial Imagery

The detection, delineation, and recognition of any significantly broad class of objects (e.g., buildings, airports, cultivated land) in aerial imagery has proven to be an extremely difficult problem. In fact, a nominal component in the solution of this problem, image partitioning, is considered to be one of the most refractory problems in machine vision.

We have formulated an optimization-based approach, applicable both to image partitioning and to subsequent steps in the scene analysis process, that involves finding the "best" description of the image in terms of some specified descriptive language.

In the case of image partitioning [Leclerc88, Leclerc89a, Leclerc89b, Leclerc89c, Leclerc89d], we employ a language that describes the image in terms of regions having a low-order polynomial intensity variation plus white noise; region boundaries are described by a differential chain code. The best description is defined as the simplest one (in the sense of least encoding length)

that is also stable (i.e., minor perturbations in the viewing conditions should not alter the description). This best description is found using a spatially local and parallel optimization algorithm that has been implemented on the Connection Machine.

A desirable processing step after image segmentation is to further simplify the resulting chain-code and polynomial description by: (1) describing the boundaries using straight lines and other more global models [Leclerc89c], and (2) grouping nonadjacent regions whose intensity variation can be more simply described by a single polynomial [Leclerc90a, Leclerc90b(*these proceedings*)].

In situations where the required image description must extend beyond that of a delineation of coherent regions, we require an extended vocabulary relevant to the semantics of the given task. Fua and Leclerc deal with the problem of boundary/shape detection given a rough estimate of where the boundary is located and a set of photometric (intensity-gradient) and geometric (shape-constraint) models for a given class of objects [Fua&Leclerc88, Fua&Leclerc90]. They define an energy (objective) function that assumes a minimal value when the models are exactly satisfied. An initial estimate of the shape and location of the curve is used as the starting point for finding a local minimum of the energy function by embedding this curve in a viscous medium and solving the dynamic equations. This *energy-minimization technique*, which evolved from a less-efficient gradient-descent approach [Leclerc&Fua87], has been implemented on the Connection Machine. It has been applied to straight-line boundary models and to more complex models that include constraints on smoothness, parallelism, and rectilinearity. In an interactive mode, the user supplies an initial estimate of the boundary of some object (which may be quite complex, like the outline of an airplane) and then, if need be, corrects the optimized curve by applying forces to the curve or by changing one of few optimization/model parameters.

Automatic recognition and delineation of important cartographic objects, such as man-made structures, from aerial imagery has been addressed [Fua&Hanson89a, Fua&Hanson89b]. The basis for the approach is a theoretical formulation of object delineation as an optimization problem; practical objective measures are introduced that discriminate among a multitude of object candidates using a model language and the minimal-encoding principle. This approach is then applied in two distinct ways to the extraction of buildings from aerial imagery: the first is an operator-guided procedure that uses a massively parallel Connection Machine implementation of the objective measure [Fua89] to discover a building in real time given only a crude sketch. The second is an automated hypothesis generator that employs the objective measure during various steps in the hypothesis-generation procedure, as well as in the final stages of candidate selection; both serial and parallel (Connection Machine) approaches are implemented.

We believe that the above techniques represent significant advances in the state-of-the-art in their respective areas of image partitioning and delineation. The implemented systems based on these techniques have been able to produce excellent results in complex situations where existing (typically local) approaches fail. Future work will emphasize the incorporation of more complex models, three-dimensional contextual information, and efficient parallel implementations.

An important issue in applications involving interactive feature extraction is the *language* available to permit effective communication between the human operator and the machine. Some of the techniques discussed above are not only useful in the analysis of imagery, but also as components in the interactive exchange of information. For example, the partitioning techniques we described above are needed to permit the human to point at something in an image and have the machine *understand* which image region is being discussed. A second needed capability is an effective way of allowing the machine to properly interpret the 3-D structure of an object depicted in a 2-D sketch provided by the human operator. In a paper included in these proceedings [Fischler&Leclerc], we describe recent work which can provide some of this desired functionality (the method discussed employs a simple optimization technique to recover the 3-D wire frame corresponding to a 2-D line drawing).

5 Object Recognition in the Natural Outdoor World

The natural outdoor environment poses significant obstacles to the design and successful integration of the interpretation, planning, navigational, and control functions of a robotic device supported by a general-purpose vision system. Many of these functions cannot yet be performed at a level of competence and reliability necessary to satisfy the needs of an autonomous robot. Part of the problem lies in the inability of available techniques, especially those involved in sensory interpretation, to use contextual information and stored knowledge in recognizing objects and environmental features. One of our goals in this effort has been to design a core knowledge structure (CKS) that can support a new generation of knowledge-based generic vision systems. A second goal is to actually construct a vision system, which employs the CKS, and has the competence to recognize objects appearing in ground level imagery of natural outdoor scenes.

5.1 Condor: A Contextual Vision System

Much of the progress that has been made to date in machine vision has been based, almost exclusively, on shape comparison and classification employing locally measurable attributes of the imaged objects (e.g., color and texture). Natural objects viewed under realistic conditions do not have uniform shapes that can be matched against stored prototypes, and their local surface properties are too variable to be unique determiners of identity. The standard machine vision recognition paradigms fail to provide a means for reliably recognizing *any* of the object classes common to the natural outdoor world (e.g., trees, bushes, rocks, and rivers). In this effort [Strat&Fischler90], we have devised a new paradigm which explicitly invokes context and stored knowledge to control the complexity of the decision-making processes involved in correctly identifying natural objects and describing natural scenes.

The conceptual architecture of the system we describe, called Condor (for context-driven object recognition), is much like that of a production system; there are many computational processes interacting through a shared data structure. Interpretation of an image involves the following four process types.

- Candidate generation (hypothesis generation)
- Candidate comparison (hypothesis evaluation)
- Clique formation (grouping mutually consistent hypotheses)
- Clique selection (selection of a "best" description)

Each process acts like a daemon, watching over the knowledge base and invoking itself when its contextual requirements are satisfied. The input to the system is an image or set of images that may include intensity, range, color, or other data modalities. The primary output of the system is a labeled 3D model of the scene. The labels included in the output description denote object *classes* that the system has been tasked to recognize, plus others from the recognition vocabulary that happen to be found useful during the recognition process. An object *class* is a category of scene features such as sky, ground, geometric-horizon, etc.

A central component of the architecture is a special-purpose knowledge/database used for storing and providing access to knowledge about the visual world, as well as tentative conclusions derived during operation of the system. In Condor, these capabilities are provided by the Core Knowledge Structure (Strat&Smith88).

Visual interpretation knowledge is encoded in *context sets*, which serve as the uniform knowledge representation scheme used throughout the system. The invocation of all processing operations in Condor is governed by context through the use of various types of context sets: an action is initiated only when one or more of its controlling context sets is satisfied. Thus, the actual sequence of computations, and the labeling decisions that are made, are dictated by contextual information (stored in the Core Knowledge Structure), by the computational state of the system, and by the image data available for interpretation.

The customary approach to recognition in machine vision is to design an analysis technique that is competent in as many contexts as possible. In contrast to this tendency toward large, monolithic procedures, the strategy embodied in Condor is to make use of a large number of relatively simple procedures. Each procedure is competent only in some restricted context, but collectively, these procedures offer the potential to recognize a feature in a wide range of contexts. The key to making this strategy work is to use contextual information to predict which procedures are likely to yield desirable results, and which are not.

Condor operates as follows: For each label in the active recognition vocabulary, all *candidate generation context sets* are evaluated. The operators associated with those that are satisfied are executed, producing candidates for each class. *Candidate comparison context sets* that are satisfied are then used to evaluate each candidate for a given class, and if all such evaluators prefer one candidate over another, a preference ordering is established between them. These preference relations are assembled to form partial orders over the candidates, one partial order for each class. Next, a search for mutually coherent sets of candidates is conducted by incrementally building cliques of consistent candidates, beginning with empty cliques. A candidate is nominated for inclusion into a clique by choosing one of the candidates at the top of one of the partial orders. *Consistency determination context sets* that are satisfied are used to test the consistency of a nominee with candidates already in the clique. A consistent nominee is added to the clique; an inconsistent one is removed from further consideration with that clique. Further candidates are added to the cliques until none remain. Additional cliques are generated in a similar fashion as computational resources permit. Ultimately, one clique is selected as the best semantic labeling of the image on the basis of the portion of the image that is explained and the reliability of the operators that contributed to the clique.

Condor has already successfully analyzed large number of photographs taken at an experimental site in the foothills behind Stanford University. Additional images have been acquired and will be processed to more fully evaluate our approach. Based on our initial experiments, and the unique architecture of of our system, we are highly optimistic about the ability of Condor to overcome many of the limitations (with respect to object recognition) inherent in the traditional machine vision paradigms.

6 Object Modeling from Multiple Images

Our goal in this research effort is to develop automated methods for producing a labeled three-dimensional scene model from image sequences. We view the image-sequence approach as an important way to avoid many of the problems that hamper conventional stereo techniques because it provides the machine with both "redundant" information and new information about the scene. The redundant information can be used to increase the precision of the data and filter out artifacts; the new information can be used for such things as filling in model information along occlusion boundaries and disambiguating matches in the midst of periodic structures.

We have developed three techniques for building three-dimensional descriptions from multiple images. One is a range-based technique that builds scene models from a sequence of range images. Another is a motion analysis technique that analyzes long sequences of intensity images. And the third technique detects moving objects from a moving platform. These techniques are briefly described below.

6.1 Building Descriptions from Sequences of Range Images

Our approach for analyzing sequences of range images is to provide the system with a wide variety of generic object and terrain representations and an ability to judge the appropriateness of these representations for particular sets of data. The variety of representations is required for two reasons. First, it is needed to cover the range of object types typically found in outdoor environments. And second, it is needed to cover the range of data resolutions obtained by a robot vehicle exploring the environment.

In this approach to object modeling an object's description typically goes through a sequence of distinct representations as new data are gathered and processed. One of these sequences might start with a crude blob description of an initially detected object, include a detailed structural model derived from a set of high-resolution images, and end with a semantic label based on the object's description and the sensor system's task. This evolution in representations is guided by a structure we refer to as "representation space": a lattice of representations that is traversed as new information about an object becomes available. One of these representations is associated with an object only after it has been judged to be valid. We evaluate the validity of an object's description in terms of its temporal stability. We define stability in a statistical sense augmented with a set of explanations offering reasons for missing an object or having parameters change. These explanations can invoke many types of knowledge, including the physics of the sensor, the performance of the segmentation procedure, and the reliability of the matching technique. To illustrate the power of these ideas we have implemented a system, which we call TraX, that constructs and refines models of outdoor objects detected in sequences of range data gathered by an unmanned ground vehicle driving cross-country [Bobick&Bolles89].

We are continuing to explore the idea of using stability to evaluate the reliability of representations. In addition, we plan to develop new explanations based on support and gravity and to explore ways to combine other types of reliability criteria with that of stability.

6.2 Building 3-D Descriptions from Monocular Intensity Image Sequences

We have developed a motion analysis technique, which we call Epipolar-Plane Image (EPI) Analysis [Bolles,Baker,&Marimont87]. It is based on considering a dense sequence of images as forming a solid block of data. Slices through this solid at appropriately chosen angles intermix time and spatial data in such a way as to simplify the partitioning problem. These slices have more explicit structure than the conventional images from which they were obtained. In the referenced paper we demonstrated the feasibility of this novel technique for building structured, three-dimensional descriptions of the world.

In later work we extended this technique to locate surfaces in the spatiotemporal solid of data, instead of analyzing slices, in order to maintain the spatial continuity of edges from one slice to the next [Baker&Bolles88]. This surface-building process is the three-dimensional analogue of two-dimensional contour analysis. We have applied it to a wide range of data types and tasks, including medical images such as computed axial tomography (CAT) and magnetic resonance imaging (MRI) data, visualization of higher dimensional (i.e., greater than three-dimensional) functions, modeling of objects over scale, and assessment in fracture mechanics.

We have also implemented a version of EPI analysis that works incrementally, applying a Kalman filter to update the three-dimensional description of the world each time a new image is received [Baker&Bolles88]. As a result of these changes the program produces extended three-dimensional contours instead of sets of isolated points. These contours evolve over time. When a contour is initially detected, its location is only coarsely estimated. However, as it is tracked through several images, its shape typically changes into a smooth three-dimensional curve that accurately describes the corresponding feature in the world.

Recently we have extended of the EPI analysis technique in two directions. The first is the modeling of

biological structures from tomographic data [Baker90]. The descriptive formalism we are developing models tissue as two-dimensional manifolds in three space. We have used this type of model to demonstrate simple versions of surgical simulation, kinematic modeling, and kinematic analysis. In the second extension we are using the temporal tracking mechanism in EPI analysis to detect and track moving objects from moving sensors. We have added evaluation routines that select key features to be tracked on moving objects. We are currently exploring techniques for constructing three-dimensional descriptions of the tracked objects.

6.3 Detecting Moving Objects from Moving Sensors

Building upon our work in motion vision and terrain modeling, we have recently begun development of techniques for detecting and tracking moving objects from a moving platform. This work is being performed jointly with the Machine Vision Group at the David Sarnoff Research Center.

Motion (in a sequence of images) provides one of the strongest cues available about the presence of a possible target in a scene. However, when a sensor is moving, everything in the image is moving. Therefore, detection of possible targets requires separating the motion induced by the movement of the sensor from the motion caused by the movement of the target. One approach to this problem, which has been developed at Sarnoff and other places, is to model the "background" image flow as a simple parametric flow field, and then use this model to eliminate image motion consistent with that flow. Any motion not consistent with the background movement is labelled as a possible moving object. Of course, such an approach fails dramatically when the simple background assumption is violated (e.g., when the terrain contains many ridges and valleys, which induce a wide variety of background image motion).

The approach we have taken is based on techniques developed at SRI [Bolles,Baker,&Marimont87] and elsewhere for constructing a three-dimensional model of a scene from the motion field computed from extended sequence of monocular images. The basic assumption underlying these techniques is that the world is static and that computations can be integrated over multiple frames to generate a stable and accurate model of the geometry of the scene. Our strategy is to apply these techniques and to examine areas in the image where the computed geometry is *not* stable. The idea is that the instabilities are caused by objects that are not static in the environment.

Of course, the computed scene geometry can be unstable if the underlying motion field computation is not stable. Such instabilities arise at occlusion boundaries, such as the top of a ridge, where accretion and deletion boundaries make a motion field computation undefined. Thus, our method for detecting moving objects also detects these occlusion boundaries. We are currently investigating the characteristics of the these different events in the space-time history in order to be able to distinguish between them.

As part of our research strategy we have tested our algorithms on both simulated and real data. (We used the Cartographic Modeling Environment to provide extensive simulation data.) The advantage of simulated data is that we know "ground truth" and therefore are in a better position to judge the competence of the algorithms than when we analyze real data. This strategy has already paid off. Our initial experiments with simulated data pointed out a serious weakness in displaying warped images to demonstrate the results of optic flow computations. At occlusion boundaries optic flow techniques locate matches (and compute flow vectors) for points that have similar greyscale values. This procedure leads to stabilized intensity images when the flow vectors are used to warp one image into another, but the flow vectors are incorrect. We are now careful to use other techniques for judging the correctness of computed flow vectors.

7 Acknowledgment

The following researchers have contributed to the work described in this report: H.H. Baker, S.T. Barnard, A. Bobick, R.C. Bolles, J. Bradstreet, O. Firschein, M.A. Fischler, P. Fua, M.J. Hannah, A.J. Hanson, Y. Leclerc, L.H. Quam, G.B. Smith, T.M. Strat, R.S. Szeliski, and H.C. Wolf.

8 References

The following recent publications result fully or in part from SRI's image-understanding research program.

1. Baker, H.H., "Building Surfaces of Evolution: The Weaving Wall," *Proc. DARPA Image Understanding Workshop*, Washington, D.C., April 1988.

2. Baker, H.H., "Surface Reconstruction from Image Sequences," invited presentation, *11th European Conference on Visual Perception*, Bristol, U.K., August 1988.

3. Baker, H.H. and R.C. Bolles, "Generalizing Epipolar-Plane Image Analysis on the Spatiotemporal Surface," *Proc. DARPA Image Understanding Workshop*, Washington, D.C., April 1988.

4. Baker, H.H., "Reimplementation of the Stanford Stereo System: Integration Experiments with the

SRI Baseline Stereo System," Tech. Note 431, Artificial Intelligence Center, SRI International, Menlo Park, California, February 1989.

5. Baker, H.H., "Surface Modeling with Medical Imagery," *Proc. NATO Advance Workshop on 3D Imaging in Medicine*, Travemunde, West Germany, June 1990.

6. Baker, H.H. and T.D. Garvey, "Motion Tracking on the Spatiotemporal Surface," *Proc. IEEE Motion Vision Workshop*, Princeton, New Jersey, October 1991, (also *in these proceedings*).

7. Barnard, S.T., "Stochastic Stereo Matching Over Scale," *Proc. DARPA Image Understanding Workshop*, Washington, D.C., April 1988.

8. Barnard, S.T., "Stereo Matching by Hierarchical, Microcanonical Annealing," *Proc. DARPA Image Understanding Workshop*. University of Southern California, Vol.II, pp. 792–796, February 1987.

9. Barnard, S.T., "Stochastic Stereo Matching on the Connection Machine," in *Proc. of Fourth International Conference on Supercomputing*, Santa Clara, California, 30 April – 5 May 1989 (also in *Proc. of DARPA Image Understanding Workshop*, Palo Alto, California, pp. 1021-1031, May 1989).

10. Barnard, S.T., "Stochastic Stereo Matching Over Scale," *Int. J. of Computer Vision*, 3(1)1989.

11. Barnard, S.T., "Recent Progress in CYCLOPS: A System for Stereo Cartography," *Proc. of DARPA Image Understanding Workshop*, Pittsburgh, Pennsylvania, pp. 439-448, September 1990.

12. Barnard, S.T. and M.A. Fischler, "Computational and Biological Models of Stereo Vision," to appear in Wiley Encyclopedia of AI, 2nd ed; (also in *Proc. of DARPA Image Understanding Workshop*, Pittsburgh, Pennsylvania, pp. 449-455, September 1990).

13. Bobick, A.F. and R.C. Bolles, "Representation Space: An Approach to the Integration of Visual Information," *Proc. of CVPR*, San Diego, California, pp. 492-499, June 1989, (also in *Proc. of DARPA Image Understanding Workshop*, Palo Alto, California, pp. 263-272, May 1989).

14. Bobick, A.F. and R.C. Bolles, "The Representation Space Paradigm of Concurrent Evolving Object Descriptions," (to appear in *IEEE PAMI* 1991).

15. Bolles, R.C., H.H. Baker, and D.H. Marimont, "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion," *International Journal of Computer Vision*, Kluwer Academic Publishers, Vol.I, No.1, pp. 7–5, June 1987.

16. Bolles, R.C. and A.F. Bobick, "Exploiting Temporal Coherence in Scene Analysis for Autonomous Navigation," *Proc. Robotics and Automation Conference*, Scottsdale, Arizona, May 1989.

17. Bobick, A.F. and R.C. Bolles, "An Evolutionary Approach to Constructing Object Descriptions," *Proc. of 5th International Symposium of Robotics Research*, Tokyo, Japan, pp. 172-180, August 1989.

18. Corby, N.R., Mundy, J.L.,Vrobel, P.A., Hanson, A.J., Quam, L.H., Smith, G.B., and T.M. Strat, "PACE — An Environment for Intelligence Analysis," *Proc. DARPA Image Understanding Workshop*, Boston, Massachusetts, pp. 342–350 (April 6-8, 1988).

19. Firschein, O. and M.A. Fischler, "AI Application of Supercomputers: The Vision Problem," *Fourth International Conference on Supercomputing*, Santa Clara, California, April 30-May 5, 1989.

20. Fischler, M.A. and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. ACM*, Vol.24, No. 6, pp. 381-395, June 1981.

21. Fischler, M.A. and O. Firschein, *Intelligence: the Eye, the Brain, and the Computer*, Addison-Wesley, Reading, Massachusetts, 1987.

22. Fischler, M.A. and O. Firschein, *Readings in Computer Vision*, Morgan-Kaufmann, Los Altos, California, 1987.

23. Fischler, M.A. and O. Firschein, "Parallel Guessing: A Strategy for High-Speed Computation," *Pattern Recognition*, Vol.20, No.2, pp. 257-263, 1987.

24. Fischler, M.A. and Strat, T.M., "Recognizing Objects in a Natural Environment: A Contextual Vision System (CVS)," to appear in *Automatic Object Recognition*, edited by Hatem Nasr, 1991, (also in *Proc. of DARPA Image Understanding Workshop*, Palo Alto, California, pp. 774-796, May 1989).

25. Fischler, M.A. and Y.G. Leclerc, "Recovering 3-D Wire Frames from Line Drawings," *in these proceedings*.

26. Fua, P. and A.J. Hanson, "Using Generic Geometric Models for Intelligent Shape Extraction," Proc. DARPA Image Understanding Workshop, University of Southern California, Vol.I, pp. 227-233, February 1987.

27. Fua, P.V. and Y. Leclerc, "Finding Object Boundaries Using Guided Gradient Ascent," *Proc.*

DARPA Image Understanding Workshop, University of Southern California, Vol.II, pp. 888-891, February 1987.

28. Fua, P.V. and A.J. Hanson, "Extracting Generic Shapes Using Model-Driven Optimization," *Proc. DARPA Image Understanding Workshop*, Washington, D.C., April 1988.

29. Fua, P.V. and A.J. Hanson, "Objective Functions for Feature Discrimination: Theory," *Proc. DARPA Image Understanding Workshop*, Palo Alto, California, May 1989.

30. Fua. P.V. and A.J. Hanson, "Objective Functions for Feature Discrimination: Applications to Semi-automated and Automated Feature Extraction," *Proc. DARPA Image Understanding Workshop*, Palo Alto, California, May 1989.

31. Fua, P.V. "Object Delineation as an Optimization Problem, A Connection Machine Implementation," in *Proc. Fourth International Conference on Supercomputing*, Santa Clara, California, 30 April - 5 May 1989.

32. Fua, P.V. and Y.G. Leclerc, "Model Driven Edge Detection," *Machine Vision and Applications*, Vol. 3, pp. 45-56, 1990, (also in *Proc. DARPA Image Understanding Workshop*, Washington, D.C., April 1988).

33. Fua, P.V., "Combining Stereo and Monocular Information to Compute Dense Depth Maps that Preserve Depth Discontinuities," in *Proc. of IJCAI 91*, Sydney, Australia, August 1991.

34. Fua, P.V., "A Parallel Stereo Algorithm that Produces Dense Depth Maps and Preserves Image Features," to appear in *Machine Vision and Applications*, 1991.

35. Fua, P.V. and P. Sander, "From Points to Surfaces," in *Proc. of the Workshop on Geometrical Methods for Computer Vision*, San Diego, California, 1991.

36. Fua, P.V. and P. Sander, "Reconstructing Surfaces from Unstructured 3D Points," *in these proceedings.*

37. Hannah, M.J., "SRI's Baseline Stereo System," *Proc. DARPA Image Understanding Workshop*, Miami Beach, Florida, pp. 149-155, December 9-10, 1985.

38. Hannah, M.J., "Test Results from SRI's Stereo System," *Proc. DARPA Image Understanding Workshop*, Cambridge, Massachusetts, pp. 740-744, April 1988.

39. Hannah, M.J., "Digital Stereo Image Matching Techniques," *International Archives of Photogrammetry and Remote Sensing*," Vol. 27-III, pp. 280-293, XVIth ISPRS Congress, Kyoto, Japan, July 1988.

40. Hannah, M.J., "A System for Digital Stereo Image Matching," *Photogrammetric Engineering and Remote Sensing*, Vol. 55, No. 12, December 1989, pp. 1765-1770.

41. Hanson, A.J., A.P. Pentland, and L.H. Quam, "Design of a Prototype Interactive Cartographic Display and Analysis Environment," *Proc. DARPA Image Understanding Workshop*, University of Southern California, Vol.II, pp. 475-482, February 1987.

42. Hanson, A.J. and L. Quam, "Overview of the SRI Cartographic Modeling Environment," *Proc. DARPA Image Understanding Workshop*, Washington, D.C., April 1988.

43. Hanson, A.J., "Hyperquadrics: Smoothly Deformable Shapes with Convex Polyhedral Bounds," *Computer Vision, Graphics and Image Processing* **44**, 191-210, 1988.

44. Hanson, A.J. and L. Quam, "A Cartographic Visualization Environment," in *Proc. of the Military Computing Conference*, Anaheim, California, pp. 233-240, May 3-5, 1988.

45. Leclerc, Y.G., and P.V. Fua, "Finding Object Boundaries Using Guided Gradient Ascent," *Proc. DARPA Image Understanding Workshop*, University of Southern California, Vol.II, pp. 888-891, February 1987.

46. Leclerc, Y.G., "Constructing Simple Stable Descriptions for Image Partitioning," *International Journal of Computer Vision*, Vol. 3, No. 1, 1989a.

47. Leclerc, Y.G., "Segmentation via Minimal-Length Encoding on the Connection Machine," *Fourth International Conference on Supercomputing*, Santa Clara, California, April 30-May 5, 1989b.

48. Leclerc, Y.G., "Image and Boundary Segmentation via Minimal-Length Encoding on the Connection Machine," Proc. DARPA Image Understanding Workshop, Palo Alto, California, pp. 1056-1069, May, 1989c.

49. Leclerc, Y.G., "The Local Structure of Image Intensity Discontinuities," Ph. D. Thesis, McGill University, Montréal, Québec, Canada, May, 1989d.

50. Leclerc, Y.G., "Simplicity of Description as the Basis for Visual Interpretation," Working Notes of the

AAAI Spring Symposion on The Theory and Application of Minimal-Length Encoding, Stanford University, Palo Alto, California, March 1990.

51. Leclerc, Y.G., "Region Grouping using the Minimum-Description-Length Principle," *Proc. DARPA Image Understanding Workshop*, Pittsburgh, Pennsylvania, September 1990.

52. Leclerc, Y.G. and A.F. Bobick, "The Direct Computation of Height from Shading," *Proc. 1991 Computer Society Conference on Computer Vision and Pattern Recognition*, Lahaina, Maui, Hawaii, June 1991, (also *in these proceedings*).

53. Lowrance, J.D., E.H. Ruspini, and T.M. Strat, "Understanding Evidential Reasoning," SRI Technical Note 501, January 1991, (also submitted to *International Journal of Approximate Reasoning*, December 1990).

54. Mundy, J., Quam, L., Strat, T. Bremner, B., Horwedel, M., and Welty, R., "The RADIUS Common Development Environment," *Proc. AIPR*, Washington, DC, October 1991, (also *in these proceedings*).

55. Quam, L. and T.M. Strat, "SRI Image Understanding Research in Cartographic Feature Extraction," *Proc. ISPRS2*, Munich, Germany, September, 1991 (also available as Tech Note 505, Artificial Intelligence Center, SRI International).

56. Smith, G.B. and T.M. Strat, "Information Management in a Sensor-Based Autonomous System," *Proc. DARPA Image Understanding Workshop*, University of Southern California, Vol.I, pp. 170-177, February 1987.

57. Strat, T.M. and G.B. Smith, "Core Knowledge System: Storage and Retrieval of Inconsistent Information," *Proc. DARPA Image Understanding Workshop*, Washington, D.C., April 1988.

58. Strat, T.M. "Decision Analysis Using Belief Functions," *International Journal of Approximate Reasoning*, Vol. 4, No. 5, pp. 391-418, September 1990.

59. Strat, T.M. "Natural Object Recognition," Ph.D. Dissertation, Computer Science Department, Stanford University, Report No. STAN-CS-91-1376, Stanford, CA, December 1990.

60. Strat, T.M. and M.A. Fischler, "Natural Object Recognition: A Theoretical Framework and Its Implementation," *Proc. IJCAI-91*, Sydney, Australia, August 1991.

61. Strat, T.M. and M.A. Fischler, "Context-Based Vision: Recognizing Objects using both 2D and 3D Imagery," accepted for publication in special issue of IEEE PAMI.

IMAGE UNDERSTANDING: INTELLIGENT SYSTEMS

Thomas O. Binford

Abstract

We have achieved significant results toward recognition of a very large class of complex objects in monocular images using methods that are rigorous and generalizable with practical impact. Inverse generalized transformational invariance (GTI) uses the original definition of generalized cylinders to infer the shape of parts of surfaces by an inverse process. Inverse GTI generates very wide applicability for methods based on invariants and quasi-invariants.

Introduction

We have achieved significant steps toward structural recognition of a very large class of complex objects in monocular images using methods that are rigorous and generalizable. We believe that these results are important for practical applications in surveillance, ATR, and cartography.

This program of research focuses on structural interpretation that builds up part/whole descriptions of observed objects in images as joints between observed GC parts. Structural interpretation is modular; we describe significant results toward each module: 1. image segmentation; estimation of extended curve discontinuities in 2D; 2. figure/ground discrimination of generalized cylinder (GC) primitive parts from background for about 10 objects, several with multiple parts. 3. estimation of 3D shape of GCs primitive parts from 2D image data for four GC parts; 4. estimation of shape of compound objects from shapes of GC parts. These methods are relevant to multi-sensor integration.

Recognition from among a moderately large set of object models is close at hand. Structured 3D object descriptions as joints between GC parts are the bases for indexing and recognition using 3D models. Nevatia and Binford demonstrated indexing and matching in ways that are significant for monocular data [Nevatia and Binford 73; Nevatia 74] They used depth data to generate coarse descriptions of 3D observed objects as search keys in a 3D object model database and demonstrated maximum likelihood matching of 3D observed model with a small subset of 3D object models obtained by indexing.

We expect rapid progress in research in development of these modules over the next two to three years. The goals of this research over five years are: 1. segmentation that is effective in complex scenes, especially outdoor scenes; 2. interpretation of complex objects in complex scenes with multi-sensor data; 3. integration of segmentation and structured interpretation in Image Understanding Environments; 4. low-complexity algorithms that demonstrate feasibility and effective performance in about 10 minutes on 20 mip workstations; algorithm analysis and architecture studies that establish feasibility of .1 second response.

This research integrates mathematical methods: 1. invariant and quasi-invariant observables derived from geometric representation (GCs); 2. geometric and physical representation as systems of constraints; 3. Bayesian networks for resolving uncertain constraints from estimation and evidential reasoning with multi-sensor data.

Segmentation

Segmentation serves goals of the entire vision system, especially of interpretation. We believe that we are at the beginning of effective and practical segmentation for moderately complex images, and that we have a road map for rapid progress toward moderate success with complex images. Here are basics for our approach to building practical and theoretically sound segmentation. What seems like a broad spectrum of imaging problems breaks down into a small set of segmentation

*This research was supported in part by a subcontract to Advanced Decision Systems, "Image Understanding Environments" from a contract to the Defense Advanced Research Projects Agency, and by a contract from NASA Ames "Nap of the Earth Navigation by Helicopter"

components based on careful analysis of physics and geometry of observation. There is much in common between segmenting depth data and segmenting intensity data. Multi-sensor segmentation is achievable. Complete segmentation is feasible, in the sense of a basis set of segmentation modules. Segmentation problems can be made well-defined. Segmentation is important to functioning of practical systems. Segmentation can be made effective and repeatable. These basics must be defended by quantitative scientific evidence.

Binford and Wang [forthcoming] have succeeded in improving the performance of a gradient-based operator, starting from a modified Canny operator. They have improved sensitivity by a factor of 4, i.e. decreased the threshold for gradient magnitude by a factor of 4. The only parameters used are measured parameters, sensor noise variance and impulse response function. Those parameters are measured from the sensor or measured from image content. Analysis has eliminated biases from transverse position that improved standard deviation of transverse position by a factor of 2 or more. Orientation estimate had a bias dependent on curvature, typically 35 degrees. That bias was reduced and then eliminated by a sequence of improvements. They defined a new form of gradient operator with three or four parameters: orientation, transverse position, transverse gradient and longitudinal gradient. In some cases, discrete pixel sampling introduces biases. They analyzed these biases of discrete sampling. Analysis in all cases included theoretical characterization, measurement on synthesized images and measurement on truth in real images which humans had characterized. They built a portable environment for experimentation and characterization in segmentation. In the end, the system demonstrated an accurate 2nd order statistical model proved on truth in real data for segmentation by this new operator. The operator made an impressive segmentation of a complex scene for a real application. They have begun to apply the methods to 3D images to address multi-sensor images.

An important part of this work is building an accurate statistical model of behavior of segmentation that is valuable in the Bayesian network for combining evidence, especially in multi-sensor problems.

This operator is effective for step edges in intensity images. It is not complete in the sense that the operator is not effective for many image features, e.g. lines or spots. The author has promoted development of a complete set of operators [Herskovits and Binford 69]. Binford and Wang have begun analysis and implementation of a complete basis. When the complete set exists, it will still be ineffective with texture. Over the next several years, the author plans to implement a simple effective segmentation for outdoor images including texture. This research has been supported in part by Nasa Ames as part of a program for Nap of the Earth navigation by helicopters (NOE).

Figure/Ground

A structural description of complex objects is a part/whole graph with GC primitive parts. The original definition of GCs by generalized translational invariance [Binford 71], more generally by generalized transformational invariance (GTI), provided an inverse method for detecting GC primitive parts. GTI generates a GC by a transform of the cross section as a function of a sweep along a space curve. Inverse GTI observes GCs (inverse GCs) by estimating the inverse transform among cross sections and swept sides of GCs, i.e. by discovering cross sections and swept sides that are related by a simple transform. The power of this method comes from the fact that complex cross sections with simple transforms, together with composition generate a very large class of complex objects (including blending and articulation). Complex cross sections can themselves be simplified as ribbons, i.e. GCs in a lower dimension that are generated by sweeping a curve along a space curve.

Inverse GTI and quasi-invariants provide powerful visual evidence. Bayesian networks implement a comprehensive mechanism for consistent and effective combination of evidence, visual and non-visual [Binford 87].

Inverse GTI correspondence relations utilize available data from surfaces and surface boundaries, including depth from stereo and motion, range measurement, orientation from shading or photometric stereo, and boundary discontinuity curves from monocular images. In single monocular images, discontinuity curves and shading are available.

GCs are generated by transforming a cross section surface by a congruence relation while sweeping the cross section along a space curve. Curves on cross sections are called parallels; curves along the sweep direction are called meridians, including limbs and edges. Most attention has been paid to observing and analyzing limbs of GCs [Nevaia and Binford 73, Sumaneweera et al 87, Ponce, Chelberg, Mann 87, Mohan and Nevatia 89]. For long, thin GCs in single, monocular images, limbs are observable with low computational complexity while cross sections at ends are small and distant; computational complexity high for ends to make correspondence by the methods described here. Parallels and meridians are complementary: observing cross sections by the methods used here has low complexity for short, fat GCs, e.g. a coin, for which limbs are difficult to make correspondence directly.

The community has neglected information about cross sections, although the author intended from the beginning to incorporate analysis of cross sections, and requested a student to begin implementation in 1980. In this work we integrate information from cross sections and limbs [Sato and Binford 92a]. For typical objects, information about both cross sections and limbs is available. Relations between cross sections are much

stronger than relations between limbs (meridians). In fact, there are very strong relations between cross sections that are the basis for rigorous correspondence.

The exxperimental results make the method intriguing and the breadth of scope of analysis is novel. Analytic results and the effectiveness of estimation methods were realized by the author. Experimental results were needed to be convincing. Experimental results were presented at various stages, at INRIA following ECCV 90 in May 1990, at the Encounter with Mathematics and Computer Vision in May 1990, at DARPA IU Workshop 1990, in a paper rejected by CVPR 91, in a paper for ISIR, January 1991, and in the DARPA-ESPIRIT Workshop for Invariants in Computer Vision, March 1991. Nevatia and colleagues have worked with limbs [Rao and Nevatia 87; Mohan and Nevatia 89]. Recently, they have made observations about "parallel symmetry" from symbolic data [Ulupinar and Nevatia 90] without experimental data. [Gross and Boult 90] show results using shading to infer properties of SHGCs.

Consider some GC subclasses. Cross section curves (parallels) are parallel in space for the class of cylinders with arbitrary cross sections, truncated by parallel planes. Parallel curves is an affine invariant, a perspective quasi-invariant [Binford 81]. Cross section curves for cones with arbitrary cross sections truncated by planes are scaled in space from a point. The scaling, i.e. the ratio of two intervals on a line, is an affine invariant and a perspective quasi-invariant. Cross section curves of SHGCs are also scaled about a single point. Limbs and edges along the swept sides of GCs (meridians) are also scaled about the axis. We have potential low complexity methods for a substantial superclass of SHGCs.

Parallels and meridians of SHGCs are self-similar under projection. For stereo and motion, curves are self-similar. From our point of view, this is the dominant use of invariants in vision, self-similarity in generic vision, not pose estimation for objects from memory. In this theme, each object is its own model. This representation and analysis of quasi-invariance guarantees that invariance is almost universal in images [Binford 91]. True invariants are few. Binford introduced quasi-invariants to extend the set of quantitative relations. Quasi-invariants are many. Our manifesto: The important properties of objects are quasi-invariant and generically observable with low computational and observational complexity (i.e. with moderate image resolution). The implications of this statement are extremely strong.

Figure 2a shows an edge image of an elbow. Figure 2b shows cross section curves of the female part of the elbow. Figure 2c shows meridians of the female part of the elbow. Correspondences were found by a Hough transform method that finds curves related by a constant scaling [Sato and Binford 92]. Figure 2d shows a set of curves found on the male part of the elbow. They are threads, a helix, outside the class of SHGCs. Note that cross sections of an SHGC truncated by planes appear non-straight except from viewpoints parallel to the truncating planes, i.e. almost everywhere. Figures 4-7 show figure/ground discrimination with other objects.

A Hough transform method is used to discover inverse GTI correspondence [Sato and Binford 92a]. For cylinders, the transform is a constant translation, parameterized by an angle and radius bounded by the image. For a scaling transform, there are four parameters, an origin with two parameters (possibly at infinity) and two scale factors for two curves. The Hough transform to discover the scaling transform is rather computation intensive and is somewhat insensitive. Many who use Hough transforms have the same complaints. The author planned initially to implement a different algorithm with low complexity.

Together cross sections and meridians define the surface of the female part. The areas covered by parallels and meridians correspond. The meridians are ribs, in fact, not limbs of the GC. The limbs of the cylinder are not visible, but in fact the limbs and the limit of the surface defined by the cross sections are defined relatively accurately by the correspondence between the two curves. Typically, relations between meridians are found from relations between cross sections.

Defining surfaces from discontinuities is an exercize in inferring 3D from 2D image curves [Binford 81]. We define connected components in space from connected components in the image, but with components in the image defined by inverse GTI, not by closed curves as in [Sumaneweera et al 88]. Note that noise and clutter have little effect on figure/ground discrimination. A large number of spurious edges from background or surface markings like text, or missing edges from failure of segmentation have a slight effect on computational complexity and a slight effect on detection of inverse relations. Thus, this method is promising for images with moderate complexity, but not for images with texture.

Estimating 3D shape

[Sato and Binford 92b, Mann and Binford 92] describe programs that build a model of parts from generic models, almost bottom-up from the image. [Mann and binford 92] also describe building the structured object model from the joint of the two part models. The resulting model is in the right form to perform indexing from the stick figure methods of [Nevatia and Binford 73].

We regard figure/ground discrimination by inverse GTI and subsequent description of primitive parts and compound objects as model-based in the sense that SHGCs or other GC subclass are generic models [Binford 82]. As in [Binford 82], generic models occur at all levels of the vision system, not just at the top level. figure 1 shows levels in the object representation of the

ultimate Bayesian net.

Cross sections of SHGCs are scaled through a point in space, hence their images are scaled through a point in the image (possibly infinite). Inverse GTI discovers a set of cross sections related by a scaling for SHGCs. For meridians, the scaling varies along the axis. A stronger condition holds than the well-known tangent condition [Ponce, Chelberg, Mann 87; Sato and Binford 92b]. Meridians contribute to estimating the 3D sweeping rule between cross sections.

[Sato and Binford 92a] analyzed about 10 examples thus far. Inverse GTI defines inverse GCs, a superset of SHGCs that are primitive parts. Inverse GCs are surfaces that must satisfy inference rules for surface interpretation that are valuable here [Binford 81]. Note that several examples are compound parts formed of several primitive parts joined.

Quasi-invariants enable estimate of sweeping rule (scaling along the axis) and cross section viewed along its normal. [Sato and Binford 92b] estimated part shape using symmetry, one method. Results are shown in figures 4 and 5. [Mann and Binford 92] hypothesize circular cylinders and helices from generic models and fit least squares solutions, shown in figures 3a and 3b.

Quasi-invariants and Probabilities

An observable is a measurement repeatable by different observers, i.e. invariant under isometries of the measurement space, or an invariant functional of observables, e.g. the distance between two identifiable points. An observable may be invariant under transforms other than isometries, e.g. perspective. There are a few perspective invariants that are useful in machine vision.

Coincidence is a true invariant but it is never observable. Incidence is an implicit assumption in interpretation of line drawings; incidence is central in interpreting surfaces from curves in monocular images. Non-coincidence is observable and quasi-invariant under rotation. Non-coincidence is stable for points until they are far enough away from the observer that their apparent distance is below the limiting resolution. If surfaces are non-coincident or non-smooth, these are quasi-invariant observables that are important for segmentation. In [Binford 87] we present a generic observability model for matte reflecting surfaces. A quick summary is that discontinuities are generic observables, i.e. observable almost everywhere.

An observable is quasi-invariant with respect to viewpoint under orthographic projection if it is constant to second order under transforms parameterized by viewing angle on the viewing sphere [Binford 81, Binford, Levitt, Mann 87]. Observables are quasi-invariant under perspective projection if they are orthographic quasi-invariants and constant to second order in the ratio of object dimension to viewing distance on the infinite viewing ball, e.g. if they are invariant under orthographic projection. Observables are quasi-invariant with respect to source orientation or location if they are constant to second order in lighting angle on the unit lighting sphere, i.e. an orthographic transform, or constant to second order in the ratio of object dimension to source distance on the infinite lighting ball, i.e. quasi-invariant under lighting perspective transforms.

Those quasi-invariants discussed thus far have been quasi-invariant under orthographic or perspective projection. These relations are much stronger than stable under general viewpoint, i.e. relations which hold on an open set about a viewpoint. Arguments about stability under general viewpoint have been used widely in analysis of polyhedra from vertices. Quasi-invariants are stronger in the sense that the domain over which they hold is large, while an open set may be infinitesimal. There is a natural structural hierarchy for generic vision based on inclusion, locality in the parameter space: properties stable under general viewpoint hold over a small support in parameter space; quasi-invariant properties hold over a large support in measure on the parameter space; generic properties hold except on a compact set of measure zero, i.e. almost everywhere. In careless notation: the domain of stable under general viewpoint is much smaller than the domain of quasi-invariant which is the same order of magnitude as the domain of generic.

Another formulation of quasi-invariants reasons about inverses under projection or observation. Here is an example of an inverse with a well-known invariant. The cross ratio is an index set for colinear 4-tuples of points in 3 space and their projections in 2D. Sets of colinear 4-tuples in 2D or 3D that have the same cross ratio form equivalence classes. The cross ratio is invariant under perspective projection from 3 space to 2 space. Thus, the cross ratio induces an inverse that is an identity of the index sets, an identity between the cross ratio in 2D and the cross ratio in 3D, and induces an identity between equivalence classes in 2D and equivalence classes in 3D, i.e. between images of colinear 4-tuples and their inverse images.

A quasi-invariant provides a correspondence between index sets for structures in 3D and 2D. The correspondence is a function of projection parameters. For invariants, the inverse is unique from 2D index sets and their equivalence classes to 3D index sets and their equivalence classes, i.e. a delta function. For quasi-invariants, the inverse is a function of projection parameters (usually unknown), approximately a delta function. A probability distribution over viewing parameters induces a distribution over the inverse of index sets in 2D to 3D and from corresponding equivalence classes in 2D to equivalence classes in 3D.

Bayesian Networks

In hierarchical vision system representation, objects are composed of parts. Figure 1 shows a current per-

ceptual hierarchy. Uncertainty is inherently conditional on context, i.e. conditional on hypothesis including viewing conditions. Physics implemented in geometry provides a basis for determining prior and conditional probabilities for relations between visual elements. All measurements have uncertainty. Almost all our constraints are uncertain constraints involving measurement. Physical constraints as uncertain geometric constraints provide conditional probabilities, the structure for an implicit Bayes net. Solution of an implicit Bayes net is inherently an inverse process. Quasi-invariants provide an implicit subnet that can be used for hypothesis generation to control evidential accrual [Levitt 88]. Quasi-invariants provide a way of inverting a local subnet, defined by an object primitive part.

Inference from images in machine vision is massively ambiguous and errorful because the evidence provided in an image relates to the appearance of an object, not the object itself. Evidence in support or denial of a given object is partial and sometimes incorrect due to noise in segmentation and/or errors in interpretation algorithms. On the other hand, there is typically an abundance of evidence. Bayesian inference provides a framework that is mathematically coherent, and thus provides a sound basis for accruing belief in support or denial of hypotheses interpreting observations as physical objects and their relationships. Evidence accrual by Bayesian networks is a natural, coherent mechanism for globally consistent interpretation of visual evidence from multiple sensors and sources with non-visual knowledge.

Probability is used to accrue belief; [Levitt, Binford, Mann 87] described initial experiments with Bayesian nets. Utility theory is used to control evidential accrual [Levitt, Binford and Ettinger 88]. Visual interpretation has a very large search space that the system cannot search exhaustively. The utility theory method has been used in practice and together with Bayesian nets, provides a comprehensive method to integrate evidence in computer vision. [Chelberg 89] discriminated objects (valve, elbow and other) from these experiments with range data using Bayesian networks based on these works.

Small parts of the entire Bayesian network were implemented here [Mann and Binford 92].

IUE: Image Understanding Environments

[Mann and Binford 92] describe a mathematically based representation for constraints. The entire SUCCESSOR effort has been oriented toward a geometric and physical basis for its type system. Typical systems have been image level with 3D range images, terrain data, Simple modeling has been added. An aim of SUCCESSOR has been to explore the level of estimation and object interpretation: to implement, incorporate, and refine evidential reasoning in the form of Bayesian networks, symbolic constraint expressions, segmentation and representation of image features and structures, geometric models (generic, abstract and specific(, and quasi-invariant reasoning about inverses from images. Considerable progress has been made along several of those aspects of high level vision.

References

[Agosta 91] Agosta, J.M.; "Probabilistic recognition networks, an application of influence diagrams to visual recognition"; Stanford University Ph.D. Thesis, August, 1991.

[Andersen 90] Andersen, S.K.; "HUGIN, a shell for building Bayesian belief universes for expert systems"; in Readings in Uncertain Reasoning, J. Pearl and G. Shafer eds., Morgan Kaufmann, San Mateo, California, 1990.

[Binford 71] T.O. Binford: "Visual Perception by Computer"; Invited paper at IEEE Systems Science and Cybernetics Conference, Miami, 1971.

[Binford and Brooks 79] Binford, T.O., Brooks, R.A.; "Geometric Reasoning in ACRONYM"; Proc Image Understanding Workshop, Palo Alto, Calif, 1979.

[Arnold and Binford 80] Arnold, R.D., Binford, T.O.; "Geometric Constraints in Stereo Vision"; Proc SPIE Meeting, San Diego, California, July 1980.

[Binford, Brooks, Lowe 80] Binford, T.O., Brooks, R.A., Lowe, D.G. "Image Understanding Via Geometric Models"; Proc 5th Int Conf on Pattern Recognition, Miami, December 1980.

[Binford 81] T.O.Binford; "Inferring Surfaces from Images"; Artificial Intelligence Journal August, 1981.

[Binford 82] T.O.Binford; "Survey of Model-Based Image Analysis Systems"; Int J. of Robotics Research, v 1, p 18, 1982.

[Binford 83] T.O. Binford; "Stereo Vision: Complexity and Constraints"; Proc First Int Symposium of Robotics Research, Bretton Woods, NH, September 1983.

[Binford, Levitt, Mann 87] T.O. Binford, T.S. Levitt, W.B. Mann; ' "Model-Based Bayesian Inference in Computer Vision"; proceedings Uncertainty in AI Workshop at AAAI 87; reprinted in Uncertainty in Artificial Intelligence III, L.N. Kanal, T.S. Levitt, R.D. Shachter and J.F. Lemmer, eds. North-Holland, 1989.

[Binford 87] T.O.Binford; "Generic Surface Interpretation: Observability Model"; Proc Int Symp on Robotics Research, 1987.

[Binford 91] T.O.Binford; "Inverse generalized cylinders: Inverse generalized Transformational Invariance and Quasi-Invariance"; Proc Darpa-Espirit Workshop on Invariance in Computer Vision, 1991

[Brooks 81] R.A.Brooks; "Symbolic Reasoning among 3-dimensional models and 2-dimensional images"; Artificial Intelligence 17,285, 1981.

[Chelberg, Lim, Cowan 84] D. Chelberg, H.S. Lim, C.K. Cowan, "ACRONYM Model Based Vision in the Intelligent Task Automation Project"; Proc DARPA

Image Understanding Wkshp, 1984.

[Chelberg 89] D.M.Chelberg; "An approach to geometric modeling using gnerealized cylinders and interpretation of range images using Bayesian Networks"; PhD Dissertation; Stanford University, 1989.

[Garvey 76] T.Garvey; "Perceptual Strategies for Purposive Vision"; PhD Thesis, EE Dept, Stanford University, 1976.

[Gross and Boult 90] A.D.Gross and T.E.Boult; "Recovery of Generalized Cylinders from a single intensity view"; Proc Image Understanding Workshop pp 557-564, 1990.

[Healey and Binford 87a] G.Healey, T.O.Binford; "Local Shape from Specularity;" Proc Int Conf on Computer Vision, London, 1987.

[Healey and Binford 87b] G. Healey, T.O.Binford; "The Role and Use of Color in a General Vision System"; Proc DARPA Image Understanding Workshop, 1987.

[Lauritzen and Spiegelhalter 88]. S. Lauritzen, D. Spiegelhalter; "Local computations with probabilities on graphical structures and their applications to expert systems"; J.royal Statistical Society G., 50 pp 157-224 (1988).

[Levitt 86] T.S. Levitt; "Model-based Probabilistic Inference in Hierarchical Hypothesis Spaces"; in Uncertainty in Artificial Intelligence, L. Kanal and J. Lemmer, eds. North-Holland, 1986.

[Levitt, Binford, Ettinger 88] T.S. Levitt, T.O. Binford, G.J. Ettinger; "Utility Theory in Computer Vision"; proceedings Uncertainty in AI Workshop at AAAI 88; *reprinted in Uncertainty in* Artificial Intelligence IV, L.N. Kanal, T.S. Levitt, R.D. Shachter and J.F. Lemmer, eds. North-Holland, 1990.

[Levitt 89] T.S. Levitt, J.M. Agosta, T.O. Binford; "Influence Diagrams in Computer Vision"; proceedings Conference on Uncertainty in AI; reprinted in Uncertainty in Artificial Intelligence V, R.D. Shachter, M.H. Henrion, L.N. Kanal and J.F. Lemmer, eds. Morgan-Kaufmann Publishers, San Mateo, California, 1991.

[Mann and Binford 92a] W.B.Mann and T.O.Binford; "An example of 3D interpretation of images using Bayesian Networks"; Proc Image Understanding Workshop 92 (to appear).

[Mann 92b] W. Mann and T.O.Binford; "Generic Modeling in SUCCESSOR: VSCP Graph and Constraint Relations"; in preparation;

[Mohan and Nevatia 89] R. Mohan and R. Nevatia' "Perceptual organization for segmentation and description"; Proc Image Understanding Workshop, pp 415-424, 1989.

[Nevatia and Binford 73] R.K. Nevatia, T.O. Binford; "Structured Descriptions of Complex Objects"; Proc AAAI 1973

[Nevatia 74] Nevatia, R.; "Structured Descriptions of Complex Curved Objects for Recognition and Visual Memory"; Artificial Intelligence Laboratory, Stanford University, Memo AIM-250, 1974.

[Pearl 86] J. Pearl; "Fusion, Propagation, and Structuring in Bayesian Networks"; Artificial Intelligence, 1986.

[Pearl 88] J. Pearl; "Probabilistic Reasoning in Intelligent Systems"; Morgan Kaufmann Publishers, San Mateo, California, 1988.

[Ponce, Chelberg, Mann 87] J.Ponce, D.Chelberg, W.Mann; "Analytical Properties of Generalized Cylinders and Their Projections"; Proc DARPA Image Understanding Workshop, 1987.

[Ponce and Healey 88] J. Ponce and G.Healey; "Using generic geometric and physical models for representing solids"; Proc Image Understanding Workshop, 1988.

[Rao and Nevatia 87] K.Rao and R. Nevatia; "From sparse 3-d data directly to volumetric shape description"; Proc Image Understanding Workshop, pp 366-369, 1987.

[Sato and Binford 92a] H.Sato and T.O. Binford; "On finding the ends of SHGCs in an edge image"; Proc Image Understanding Workshop, 1992.

[Sato and Binford 92b] H.Sato and T.O. Binford; "BUILDER-I: a system for the extraction of SHGC objects in an edge image"; Proc Image Understanding Workshop, 1992.

[Ulupinar and Nevatia 90] F. Ulupinar and R. Nevatia; "Recovering shape from contour for SHGCs and CGCs"; Proc Image Understanding Workshop, pp 544-566, 1990.

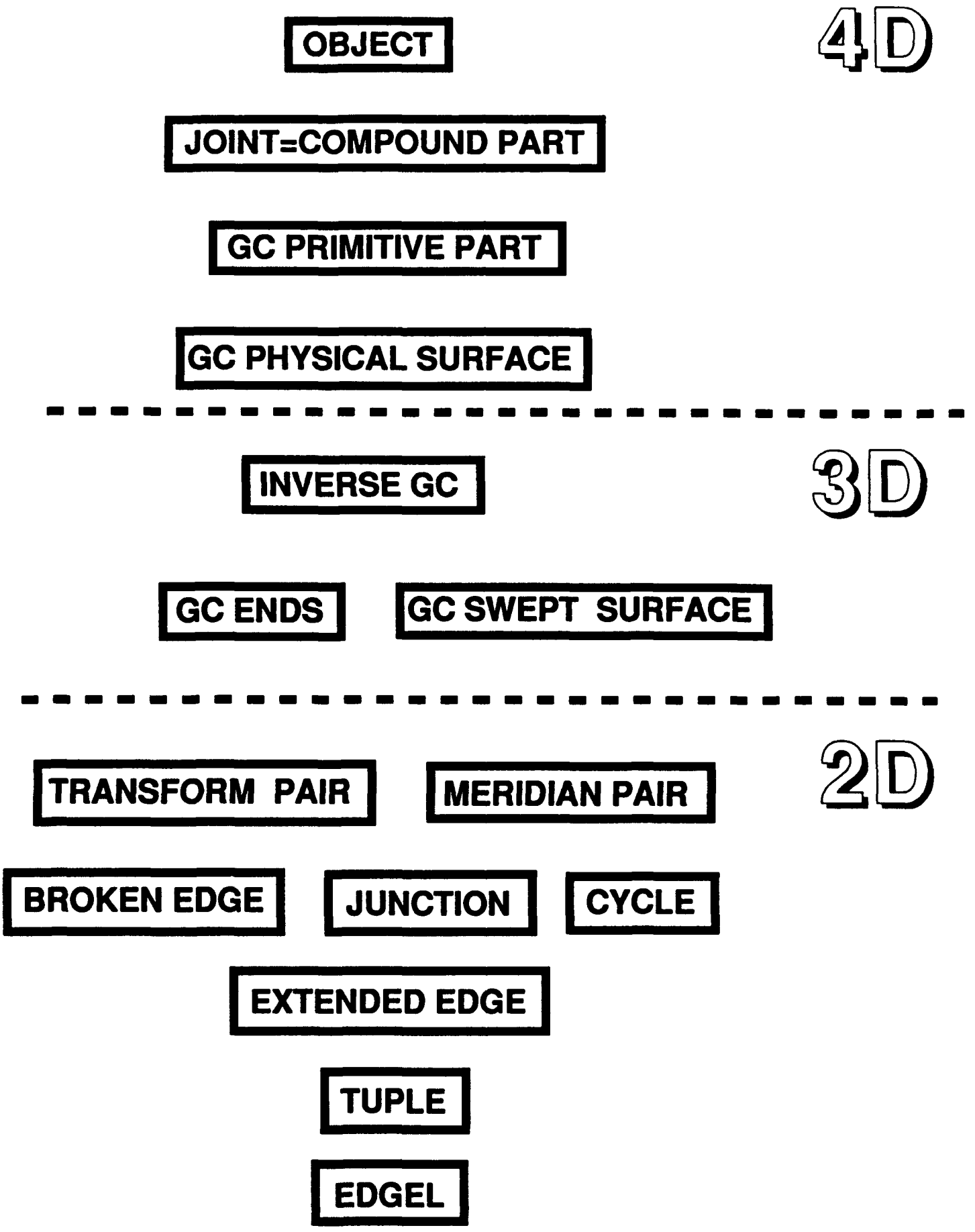
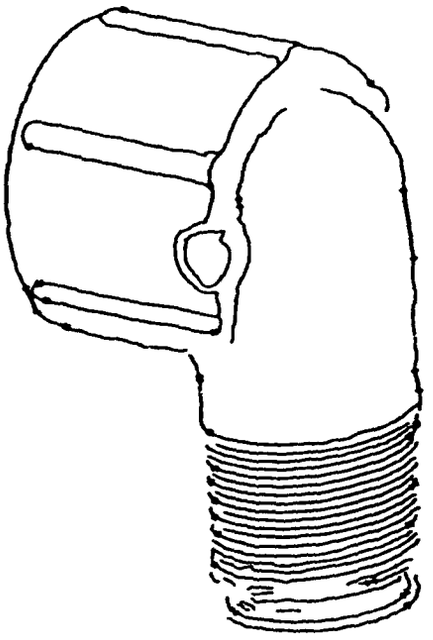


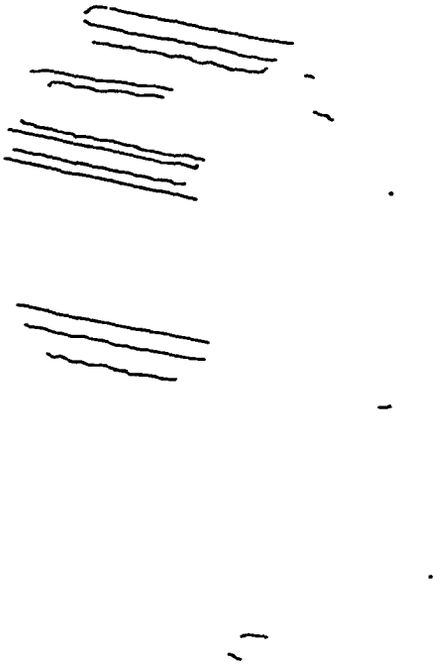
Figure 1: Perceptual Hierarchy DAG for Object Interpretation including monocular images.



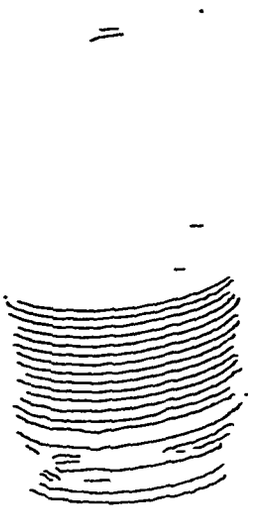
2a. Edge image



2b. Corresponding edges from cross sections of female cylinder from elbow



2c. corresponding meridians from female cylinder from elbow



2d. corresponding edges from cross sections of male cylinder from elbow

Figure 2: Segmentation into GC primitive parts of elbow joint image

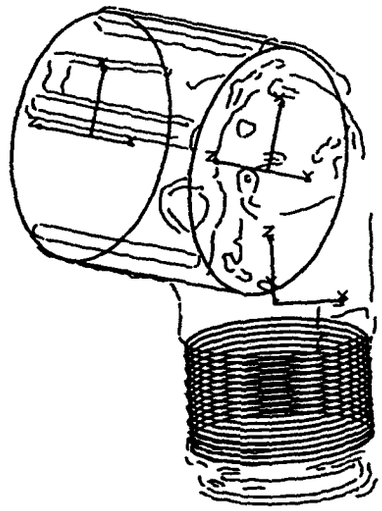


Figure 3a. Final fit of a cylinder and helix to the image data.

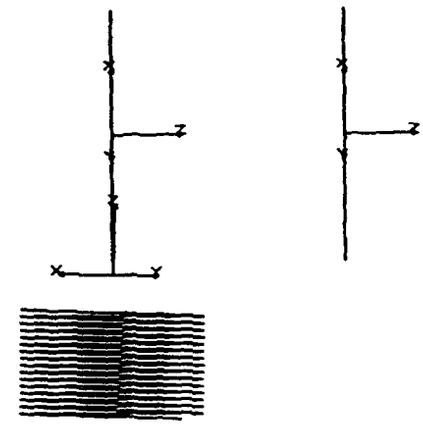
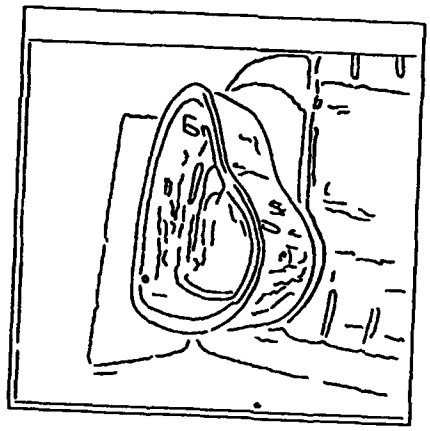
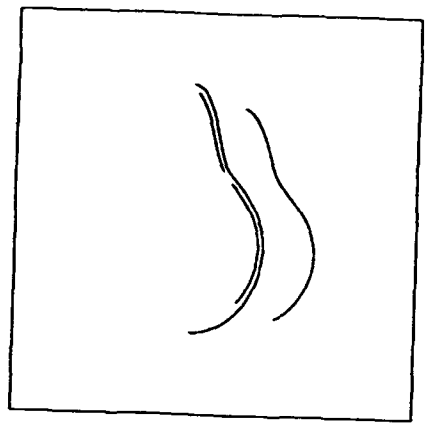


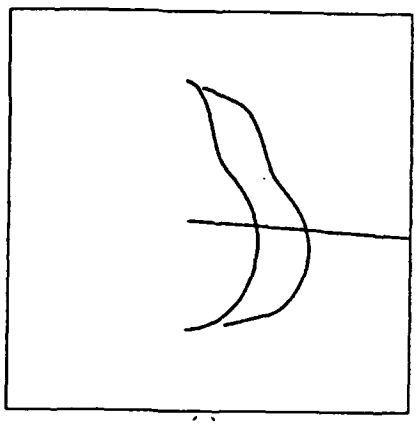
Figure 3b. Model rotated for side view.



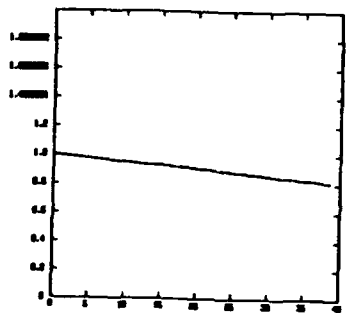
4a. edge image



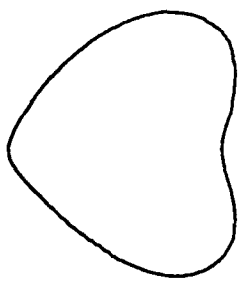
4b. cross sections



4c. cross sections with meridians

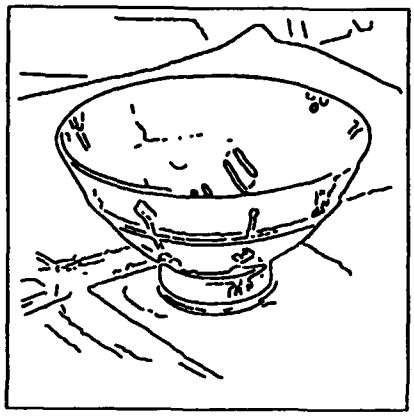


4d. scaling; sweeping rule;

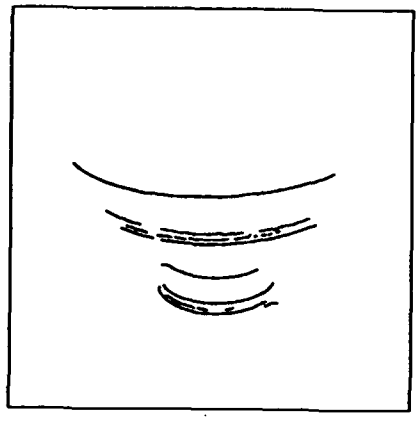


4e. cross section viewed along normal;

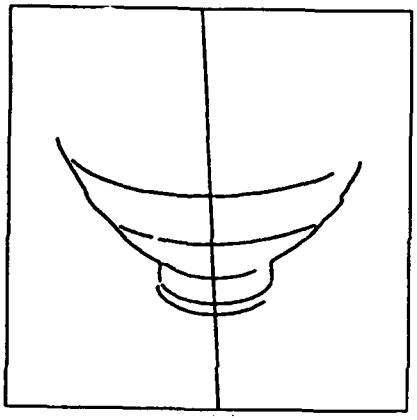
Figure 4. Figure/Ground discrimination and 3d shape estimation for Heart Shaped Cup.



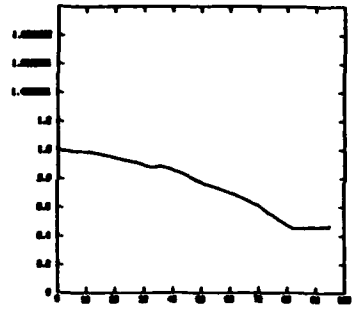
5a. edge image



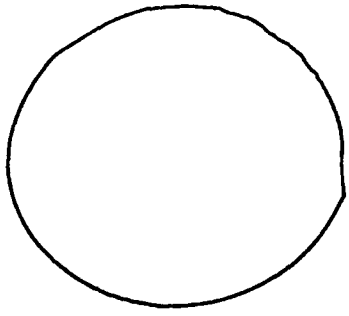
5b. cross sections



5c. cross sections with meridians

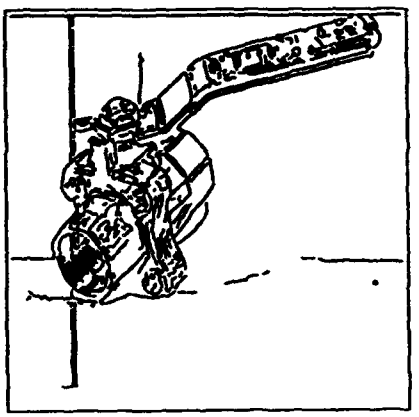


5d. scaling; sweeping rule;

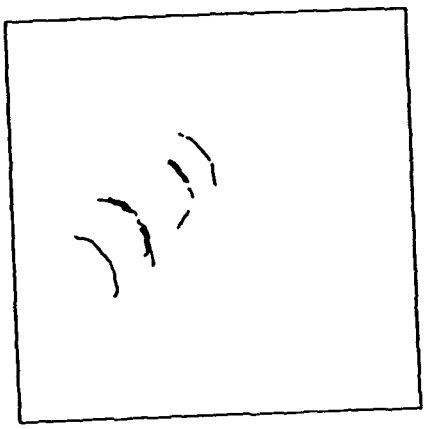


5e. cross section viewed along normal;

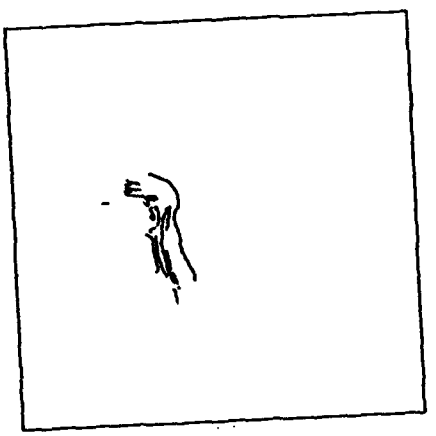
Figure 5. Figure/Ground discrimination and 3d shape estimation for Rice Bowl.



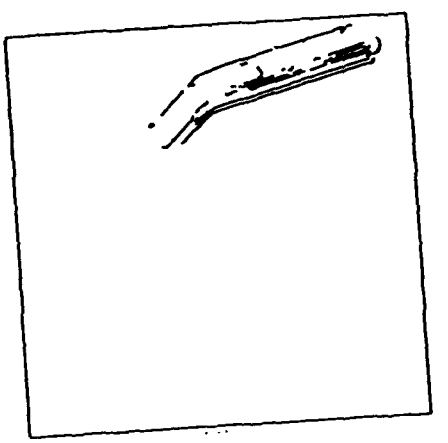
6a. edge image



6b. cross sections of one part

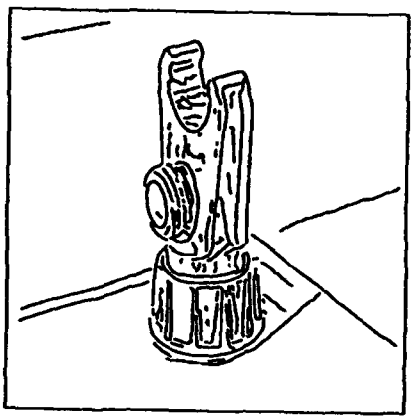


6c. cross sections of another part

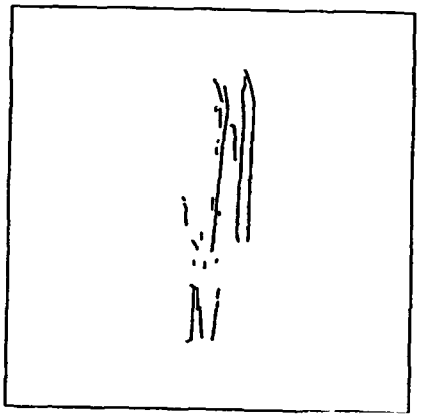


6d. cross sections of another part

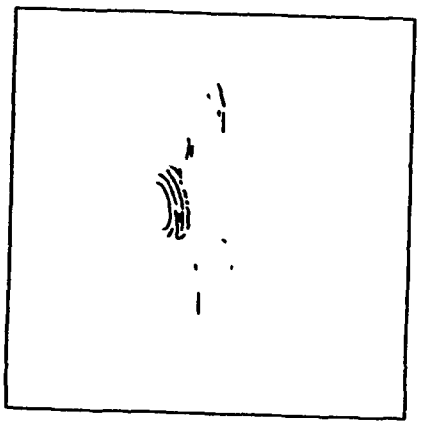
Figure 6. Figure/Ground discrimination for Valve.



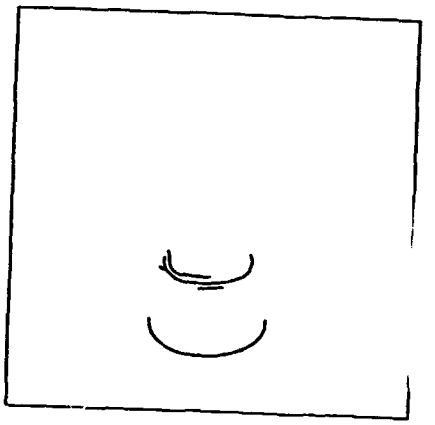
7a. edge image



7b. cross sections of one part



7c. cross sections of another part



7d. cross sections of another part

Figure 7. Figure/Ground discrimination for SHGC.

CMU Image Understanding Program

Takeo Kanade
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Image Understanding research at CMU addresses a broad spectrum of issues, from applications of machine vision to the development of new sensors and basic vision science. The focal areas of our research include:

- *Vision for Site Modeling*
- *Physics-Based Vision*
- *Parallel Vision*
- *Sensor Development*
- *Vision for Object Recognition and Manipulation*
- *Vision for Robot Vehicles*

1 Vision Methods for Site Modeling

Rapid site modeling – the generation of a detailed three-dimensional model of a surveyed site – is a critical problem for both military and civilian applications, including geographic database generation for cartography, reconnaissance, damage assessment, combat simulation, and autonomous air/ground vehicle navigation.

Central to the site modeling problem is the development of efficient and reliable Image Understanding techniques to analyze and extract precise three-dimensional shape information from multiple visual or range images taken from a moving platform, such as a scouting ground/air vechicle, or a stereoscopic camera. We have recently developed three new techniques: 1) the factorization method for shape and motion recovery from an image sequence; 2) the multi-baseline stereo method for reliable and dense depth mapping; and 3) the landmark object modeling method from range image sequence. All of these methods have been tested with images taken in a controlled laboratory environment (to provide ground-truth data for quantitative evaluation) and taken outdoors under real lighting and geometry conditions. Their performance has been demonstrated to exceed that of previous methods.

1.1 The Factorization Method

The structure from motion problem – recovering scene geometry and camera motion from a sequence of images – has attracted much of the attention of the vision community over the last decade, and yet it is common knowledge that existing solutions work well for perfect images, but are very sensitive to noise There are two fundamental reasons for this. First, when camera motion is small, effects of camera rotation and translation are conjugate: for example, rotation about the z-axis and translation along the x-axis both generate a very similar change in an image. Any attempt to recover or differentiate between these two motions is naturally noise sensitive. Second, computation of shape as relative depth, for example, the height of a building as the difference of depths between the top and the bottom, is very sensitive to noise, since it is a small difference between large values. These difficulties are especially magnified when the objects are distant from the camera relative to their sizes, which is usually the case for interesting applications such as site modeling.

Recently we (Tomasi and Kanade) [1][2] observed that both difficulties disappear when the problem is reformulated in world-centered coordinates unlike the conventional camera-centered formulation. This new formulation links object-centered shape to image motion directly, without using retinotopic depth as an intermediate quantity, and leads to a simple and well-behaved solution. Furthermore, the mutual independence of shape and motion in world-centered coordinates makes it possible to cast structure-from-motion as a factorization problem, in which a matrix representing image measurements is decomposed directly into camera motion and object shape.

More specifically, an image sequence can be represented as a $2F \times P$ measurement matrix W, which is made up of the horizontal and vertical coordinates of P points tracked through F frames. If image coordinates are measured with respect to their centroid, we prove the following *rank theorem*: under orthography, the measurement matrix is of rank 3. As a consequence of this theorem, we show that the measurement matrix can be

factored into the product of two matrices M and S. That is, $W = MS$, where M is of size $2F \times 3$ and encodes camera motion, and S is a $3 \times P$ matrix which encodes shape in the coordinate system attached to the object centroid.

The rank theorem precisely captures the nature of the redundancy in an image sequence, and permits a large number of points and frames to be processed in a conceptually simple and computationally efficient way to reduce the effects of noise. The resulting algorithm is based on Singular Value Decomposition, which is numerically well-behaved and stable. The robustness of the recovery algorithm in turn enables us to use an image sequence with a very short interval between frames, which makes feature tracking relatively easy.

We have demonstrated the accuracy and robustness of the method in a series of experiments on laboratory and outdoor sequences, with and without occlusions. With a laboratory experiment where a camera is moved by means of a high-precision positioning platform for collecting the ground-truth data, it was shown that the computed motion has errors consistently less than 0.1 degrees. The computed motion preserves all discontinuities in the rotational velocities. The errors in shape recovery were found to be less than 1 %.

The method was also tested with image sequences of outdoor scenes taken by a hand-held camcorder. Outdoor images are harder to process than lab images because of unpredictable lighting changes and camera "jaggle". Also, in real applications features can appear and disappear from image to image due to occlusions, which produces a W with incomplete columns. However, we can still apply the method successfully. The paper by Tomasi and Kanade in these proceedings presents the theory and experimental results of the factorization method.

1.2 Multi-Baseline Stereo

Much progress has been made in methods for stereo vision to reconstruct the three-dimensional world from images taken from slightly different points of view. However, a fast, reliable stereo vision system for general vision use remains an unrealized goal of computer vision. The main difficulty is still the correspondence problem – to get high precision, the cameras should be far apart, but then it is difficult to match corresponding points.

One of the most common methods used to deal with the problem is a coarse-to-fine control strategy, in which coarse resolution matching removes false matches and high resolution gives a precise depth value. Using the coarse-fine strategy, however, does not always remove false matches, especially when there is inherent ambiguity in matching such as a repeated pattern over a large region (eg., a scene of a picket fence).

We (Kanade and Okutomi) [3] developed a new technique called Multi-Baseline Stereo which uses multiple images obtained by cameras that are laterally displaced (either or both horizontally and vertically) to produce different baselines. The technique shares features with the trinocular stereo, the EPI method, and Kalman-filter based methods in that it utilizes multiple images to increase the accuracy and precision of matching, but the computational approach is very different. Previous methods obtain intermediate candidate matchings over multiple stereo images and check the consistency among these to find the correct combinations with increased precision. Since the intermediate decisions on correspondences are inherently noisy and ambiguous, finding the correct combinations requires sophisticated consistency checks and search or filtering.

In contrast, the Multi-Baseline Stereo uses the simple fact that for any baseline, disparity divided by the baseline distance is constant for each point.

$$\frac{d}{B} = F\frac{1}{z} = \zeta. \qquad (1)$$

Therefore if we represent evidences of matching from individual stereo image pairs with respect to ζ, rather than the disparity d as is usually done, then they should show indivate consistently the correct matching position. We may then add, integrate, or fuse such evidences of matching so that the resultant integrated evidence uniquely determines the correct match.

We have implemented this idea by using the SSD (sum of squared differences) over a small window as the simplest and most effective measure of matching . We represent the SSD values from individual stereo pairs with respect to the inverse distance 1/z. The resulting SSD functions from all stereo pairs are added together to produce the sum of SSDs, which we call *SSSD-in-inverse-distance*. We have proven that the SSSD-in-inverse-distance function exhibits a unique and clear minimum at the correct matching position even when the underlying intensity patterns of the scene include ambiguities or repetitive patterns.

How this method works can be best illustrated by the example in figure 1 which shows two of the ten images of a scene. The top part of the image (grid pattern) is completely repetitive. So, the matching is inherently ambiguous for a point in that region. Figure 2 shows the SSSD-in-inverse-depth for such a point. The bottom curve which is obtained by a single baseline shows multiple minimums. We can observe, however, that as the number of baselines increases to two, four and eight, the SSSD-in-inverse-distance has a better-defined minimum. The computation is completely local, and does not involve any search, optimization, or smoothing.

The method has been tested with both indoor and outdoor scenes.The paper by Kanade, Okutomi, and Nakahara in these proceedings presents the theory and experiment that we have produced so far.



(a) (b)

Figure 1: "Town" data set: (a) Image0; (b) Image9

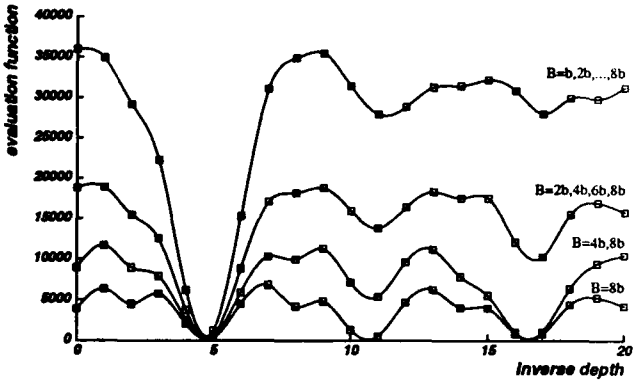


Figure 2: Combining multiple baseline stereo pairs

1.3 Landmark Modeling from Range Image Sequence

We (Hebert) have been developing techniques [4] to map the environment of a mobile robot using a laser range finder. We have extended the map building techniques to build explicit three-dimensional models of the landmarks and to use those models for navigation by matching the models with observed object models [5]. Including explicit shape models allows for selective landmark identification and for a more detailed map of the environment.

The object models are built by gathering range data from several closely-spaced images. The features and data collected on each object are used to fit a discrete surface, represented by a mesh of points, which constitutes the object model stored in the map. This approach does not involve an explicit segmentation of the observed scene. Instead, features extracted from individual range and reflectance images are grouped into clusters corresponding to objects in the scene. The features are range and reflectance edges and near-vertical regions. Each cluster is assumed to correspond to one object. Clusters are tracked from image to image using a previously developed matching algorithm. For each object, the surface fitting process is iterated using data and features from the clusters in the new and previous images.

We have implemented and tested these algorithms using the Navlab and Navlab II as testbed vehicles with the Erim laser range finder. The Perceptron range finder, which has better spatial resolution, was also used for off-line experimentation. Models of natural and man-made objects were successfully built and matched with observations during vehicle travel, yielding correct vehicle position. In our experiments so far, the perception system was tested in isolation, gathering image and position data, building and matching models without actually sending driving commands or position corrections to the vehicle. Our goal is now to incorporate the algorithms in the existing Navlab navigation environment to demonstrate improved mission capabilities.

2 Physics-Based Vision

It has long been recognized that traditional feature detection such as edge-finding acts in ignorance of the optical physics of vision, and thus provides an unreliable foundation for machine vision. To address this problem, The CMU Image Understanding group has been spearheading physics-based vision that performs an explicit analysis of the constraints of optics, physics, geometry, and uncertainty involved in vision [6]. We continue to make progress in this area for the analysis of appearance including color and interreflection, surface roughness, texture, photometric stereo, and image acquisition.

2.1 Visual Analysis of Appearance

To date, physics-based vision research has primarily been limited to trivialized scenes with a single surface of relatively simple properties, analyzed under fairly ideal conditions. This is a far cry from the analysis of the complex scenes found in real-life situations, with mixtures of materials, shapes, colors, and textures, uneven illumination, and many objects present in juxtaposition.

Clearly, such complexity is beyond our ability to describe and analyze at present. However, this analysis of the appearance of objects in arrangements is a necessary evolution from the study of physics-based vision in idealized scenes.

Understanding Interreflection Between Objects

The computer graphics community has modeled interreflection for some time, but it presents very formidable problems for machine vision. This is because interreflection can take so many forms. In one extreme case, reflection from a mirror can take the form of the appearance of nearby objects. Recent work in machine vision has addressed the analysis of interreflection on a single concave surface; and the detection (but not analysis) of interreflection among multiple colored surfaces.

In our work, we (Novak and Shafer) address the understanding, modeling, and analysis of interreflection among

multiple objects. Even if we restrict ourselves to dielectric materials that obey the Dichromatic Reflection Model [7][8], the analysis is very difficult; to provide sufficient information for analysis, we are studying color images. According to the Dichromatic model, the color histogram for a single surface will form a "skewed T" shape within a plane, with a baseline corresponding to the diffuse color of the surface and a branch in the direction of the color of the highlight (i.e. the illumination color). We have shown that interreflection between two surfaces causes this histogram to have a more complex shape, which will not generally be planar. In the area of the interreflection, there are several components due to body/body, surface/body, body/surface, and surface/surface interreflection. These components interact in a highly structured manner to modify the distribution in the color histogram.

The regularity of these structures in the histogram, however, forms a basis for analysis of surface properties. For example, the main highlight on the object corresponds to the main branch of the color histogram. The length of this branch is inversely related to the roughness of the surface, because a smooth surface concentrates the highlight into a narrow and hence very bright spot, while a rough surface diffuses the highlight and thus produces a shorter branch in the histogram. Also, the intersection of this branch with the branch for the (diffuse) body reflection color will be broad when the surface is rough and the highlight is spread over a broad patch of the surface. If the surface is smoother, the intersection of these branches will be narrow because the highlight is concentrated over a small spot on the object surface. The point at which these branches meet corresponds to some amount of body reflection, which in turn tells the angle of illumination on the surface.

In the case of interreflection, there are many more structures on the histogram. For example, the surface/surface reflection causes a "secondary highlight" on the surface, which corresponds to a new branch present in the histogram. This has no counterpart when looking at a single surface without interreflection. The secondary highlight is very revealing, because its length, angle, and position all are related to the roughness of the two surfaces involved.

Other structures in the color histogram under interreflection reveal information about the geometric relationship of the surfaces and the light source, and even about the relationship of the reflectance spectra of the surfaces' body reflection. Our work in this area is now aimed at demonstrating each of these analyses of the color histogram and determining the reliability and usefulness of each. To the extent that we succeed, we may be able in the future to analyze and understand interreflection of colored objects and overcome the confusion suffered by current methods for color image segmentation.

Determining Surface Roughness from Reflection

When a person looks at an object, or an image of an object, it is easy to determine in qualitative terms whether the object's surface is smooth or rough. The primary source of this information is the sharpness or blurriness of the edges of reflections seen on that surface. If the surface is smooth, reflections seen there will be very sharp-edged and clear, as in the commercial where the woman holds up a dinner plate fresh out of the dishwasher and exclaims, "I can see myself!" On the other hand, if the surface is rough or dirty, then reflections seen on that surface will have fuzzy edges and be indistinct. This is the aspect of appearance sometimes called "distinctness-of-image gloss".

We (Stone and Shafer) are studying how to analyze this property in real images to estimate surface roughness. We begin by noting that the most revealing aspect of surface reflections is the blurriness of a reflection of a step edge. For example, if there is a light source in the scene, its reflection is apparent as a highlight on the surface. Frequently, other objects or secondary light sources (such as windows) cast light onto a surface, whose reflection similarly reveals the roughness of the reflecting surface.

In terms of mathematical physics, these situations are all the same – a source with a step edge casting light onto a reflecting surface that reflects the light into the camera. The roughness of the reflecting surface in effect convolves the incident step edge with some blurring function; this gives rise to the appearance of a blurred reflection edge in the image. We have developed a mathematical characterization of this convolution based on the Torrance-Sparrow reflection model, which is very widely used and accepted. By identifying the edge of the reflection and analyzing intensity profiles across that edge, we can estimate the probability distribution function of the surface microfacet normals in the surface microstructure.

To date, we have analyzed only the simplest arrangement of a very large source with a single clearly defined step edge, reflecting from a planar surface into the camera. We have found diffraction gratings, under particular conditions, to be useful as a test object, since the surface facet normals distribution is well-known. Our ongoing work will attempt to generalize all of these assumptions. Our goal is to be able to look into a complex scene, identify a reflected pattern on a surface, and by analyzing its blurriness or sharpness, compute the roughness of that surface.

Analyzing Surface and Image Texture

3D texture analysis has been a difficult area for machine vision, because a texture by definition has a complex geometry in the image. A good way to characterize texture is by the Fourier transform, but unfortunately this destroys inherent geometric information. Instead, we use a form of local Fourier transform called the image

spectrogram, which is a kind of dense wavelet decomposition of the image. In the spectrogram, a local Fourier transform is computed in a neighborhood around each pixel in the image. Thus, the spectrogram is S(x,y,u,v) where x and y are image coordinates and u and v are spatial frequencies. The spectrogram allows a unified treatment of geometry, Fourier transforms, wavelets, and image resolution and sampling, which makes it particularly useful for analyzing 3D image textures. We have investigated other similar representations including Gabor functions, orthonormal wavelets, steerable filters, and the Wigner distribution. We find the image spectrogram to be superior for 3D machine vision because it gives a dense representation of space/frequency information and allows all the mathematics of Fourier transforms to be brought to bear for geometric image analysis.

Our primary study (Krumm and Shafer) has been on the analysis of texture gradients on 3D surfaces by using the spectrogram. We published in the 1990 DARPA IU Workshop proceedings our theory that unified many important image phenomena including texture gradients, aliasing, and lens parameters [Krumm91]. Now, we have expanded our work from one-dimensional to two-dimensional signals (images) and developed a working algorithm for computing shape from texture.

Our new shape-from-texture algorithm exploits the systematic changes in apparent frequency in the image of a textured shape. On a plane viewed under perspective, the spatial frequencies of the texture increase as the plane recedes from the viewer. We compute the local Fourier transform at various points on such a plane. These transforms are related by an approximately affine transformation of the u-v frequency axes; the parameters of that affine transform are determined by the orientation of the plane. By computing the affine transform to match the local Fourier transforms, we can calculate the slope of the plane. This method requires no feature detection; it works on relatively small neighborhoods, which should allow us to extend it for curved surfaces.

2.2 Specular Lobe Objects - Four Light Photometric Stereo

The brightness of a pixel in an image results from the reflection of light. The amount of light reflected depends on characteristics of the reflecting surface, and imaging geometry. Two types of reflectance models have been used to predict image brightness: geometrical optics models and physical optics models. Geometrical optics models are approximate models. They are appropriate when the wavelength of light is much smaller than the roughness of the reflecting surface, and are much simpler than physical optics models. A unified reflectance model was proposed by Nayar, Ikeuchi, and Kanade [9], which combines the geometrical and physical models. This model is able to predict the brightness of surfaces that exhibit a diffuse lobe, specular lobe, and specular spike.

By using the unified reflectance model, we developed a photometric sampler device [10] that determines the surface orientation and reflection parameters of surfaces exhibiting a linear combination of lambertian and specular spike components. A photometric sampler device uses a series of extended light sources to sequentially illuminate the object under inspection.

Another interesting class of surfaces is those which exhibit a linear combination of lambertian and specular lobe components. We (Solomon, and Ikeuchi) have developed an algorithm that extracts the surface orientation and reflectance parameters for this class of objects. The method, which we call the*four light photometric stereo*, uses four lights to sequentially illuminate the object under inspection. The lights are positioned so that the specular lobes of each light source do not intersect. The four lights produce three types of regions on an object: regions illuminated by all four lights, regions illuminated by three lights, and regions illuminated by only two lights. Each of these regions provides different information.

For each region an algorithm has been developed to extract the shape and roughness of objects that exhibit a specular lobe. The work extends photometric stereo techniques for specular lobe objects, from the regions illuminated by all four light sources to the entire gaussian sphere. We have tested our algorithm with real specular lobe objects, including a specular painted sphere, and a plastic helmet.

2.3 Research in Image Acquisition

To carry out studies in physics-based analysis of images, it is important to have high-quality image data. Acquiring such data demands high-precision imaging equipment and careful calibration, so that errors in experimental results can be properly attributed to limitations of equipment, calibration, physical modeling, or computation.

We (Willson and Shafer) have just completed the construction of a second automated camera system for the Calibrated Imaging Lab. The new system consists of a motorized 13x zoom lens and a cooled slow-scan scientific camera. The new lens features 11,100 steps of resolution for focal length (zoom), 4,000 steps for focus distance, and 2,700 steps for aperture. The new camera features 12 bits of dynamic range, direct digitization of each cell of the sensor element, a 400:1 signal-to-noise ratio, and fully variable exposure times. We are presently doing the radiometric and geometric calibration of the camera system.

We are now studying range-from-focus, and have made progress in Constant-Magnification Focusing and understanding spatial aliasing in cameras. Constant-Magnification Focusing deals with the problem of focus magnification, the slight change in image magnification that occurs when a lens is focused [Willson91]. This

can disturb the calculations of traditional focus-quality measures, resulting in errors in range-from-focus computations. With Constant-Magnification Focusing, we perform a very small zoom adjustment as the lens is focused, in order to maintain a constant image magnification. This eliminates the bias caused by focus magnification.

3 Parallel Vision

The big news this year is the arrival of the first *iWarp* computers from Intel. We have received four iWarp computers, three 8x8 arrays and one 8 cell array (which will be expanded to 16 cells). The arrival of these machines is the culmination of years of joint work between Carnegie Mellon and Intel in the design of the iWarp chip and its associated software.

iWarp is explicitly designed for use in signal and image processing; it incorporates a very high per-chip I/O rate (320 MB/s total over eight physical pathways) with high performance (20 MFLOPS and a comparable number of MIPS), as well as a fast memory interface (160 MB/s). The high I/O rate is essential for signal and image processing operations, in which the ratio between computation and I/O is much lower than for scientific computing. The iWarp chips can communicate either through conventional message-passing, or through systolic I/O in which words of data are fed to the pathway directly from the computational unit, with extremely low latency. Non-adjacent chips can communicate through long-distance connections mediated by the hardware of intervening cells.

Our work on parallel computer vision has led to the commercial support of the Apply programming language [11] by Intel for iWarp. Apply allows the programmer to easily write any local image processing operation (such as point operations, edge detection, and smoothing) and have it automatically mapped efficiently onto iWarp, without programmer knowledge of details such as data distribution and interprocessor communication. Apply comes with a large library of image processing programs called the WEB library, originally developed on the Carnegie Mellon Warp machine. Apply and iWarp are already being used in a number of labs, including General Electric Aerospace.

A successor to Apply has been developed that allows both local and global image processing operations to be written [12][13]. This new language, Adapt, supports all Apply operations as well as such operations as histogram, image warping, two-dimensional fast Fourier transform, and connected components. The language has been implemented on iWarp, and is currently in use at Carnegie Mellon for research in feature extraction for a vision algorithm compiler, spectrogram generation, and multi-baseline stereo. In addition, an implementation of the second DARPA Image Understanding benchmark is under way.

One of the more exciting applications of Adapt is to overcome the image I/O bottleneck in the current configuration of iWarp. At present, iWarp is attached to the VME bus of a Sun, which limits the I/O that can be done to iWarp to a few megabytes per second – too slow for real time image processing. However, recently image compression chips and boards based on the JPEG image compression standard have come on the market, which can compress images at video rate by as much as 28:1 without loss of image quality. At that compression rate, even full 512x512 color images could be fed across the VME bus at 30 frames/second. We are presently implementing the JPEG algorithm on iWarp, which will allow images to be uncompressed and processed there, then recompressed for display by another compressing frame buffer. This is the first application of image compression to overcoming the traditional parallel computer I/O bottleneck that we are aware of.

The development of the JPEG standard has created great opportunities for the application of imaging. A new emerging image processing library standard may well have similar impact. For several years, a group of image processing experts (primarily from companies with an interest in imaging such as Datacube, Sun, Mitre, Hewlett-Packard, and so on) has been defining a standard imaging model and program library for image processing. This work, now called the Programmer's Imaging Kernel System, has recently been proposed as an ISO standard. Carnegie Mellon and Intel are jointly developing an Adapt implementation of this standard. It is believed that this standard will serve as more than just a program library for iWarp; as iWarp can easily be implemented on a variety of MIMD parallel computers (an implementation on the Touchstone is underway) it should serve as a basis for developing architecture-independent image processing systems and for comparing and evaluating new high performance computers for image processing.

4 VLSI Smart Sensor-based Range Finder

We (Gruss, Carley and Kanade) have been developing a high-speed light-stripe range-imaging system based on an analog VLSI smart sensor, which is capable of acquiring range frames at rates two orders of magnitude better than currently available ranging methods [14][15]. At the heart of this system is an intelligent VLSI sensor which uses a novel cell-parallel approach to stripe-based range imaging. The cell-parallel algorithm we employ is practical only because one can integrate computation and memory functions in a single photoreceptive cell.

In a conventional light-stripe system, a complete range map is obtained via the step-and-repeat process of projecting a stripe, grabbing a video image, extracting the stripe position from the image, and stepping the stripe until an entire scene has been scanned. Though practical, the speed of sampling range data with this conventional

technique is severely limited to the order of a second. Also, the time required to build a range map in this fashion is proportional to the desired horizontal resolution.

The new sensor is based on the same optical and geometrical principle, but it operates somewhat differently. In contrast, our range-finder acquires range data while a stripe is swept continuously over the scene. A specialized VLSI sensor consists of a two-dimensional array of smart photosensitive cells, each of which has circuitry that detects and remembers the *time* at which it observes the peak incident light intensity during a sweep. A given cell predefines a unique line of sight, and the "timestamp" it records determines a particular orientation of the stripe. Thus, sensing elements, working in parallel, gather information sufficient to extract a complete range map from a single pass of the light stripe. The time required to build the range map is independent of the map's spatial resolution.

Analog signal processing is well suited for performing the kinds of computations necessary to achieve intelligent sensors. Integrated circuitry which performs computations on analog values is small in area. The density which with the sensing and processing task can be performed within a given die area is thus enhanced. Analog processing circuits do not generate the switching noise associated with digital implementations. Finally, sensed data is by its nature analog. Analog processing avoids the need to convert sensed data before computations can take place.

Currently, a prototype cell-parallel range-imaging system, based on a smart VLSI sensor, is operational. The sensor in this prototype system is a full-custom integrated circuit implemented in a 2 μm CMOS process. The chip contains 896 smart cells arranged in a 28 × 32 array and measures 7.8 mm × 9.2 mm. Within each smart cell is a photoreceptor, light-stripe detection circuitry, and the means to record and read out range data samples. The prototype range imaging sensor and system acquires range images at rates up to 1,000 frames per second: see figure 3. Repeatability of the range data has been measured to be on the order of a percent.

A second generation light-stripe sensor implementation is currently being tested. This new chip incorporates several advantages over the first design. The die area of smart cells in the new chip is 36% smaller than that of the cells of the first generation sensor. Stripe detection is done in a more robust manner and range data readout circuitry has been simplified. In addition, the new cell provides a means to record the intensity of the light-stripe measured when range data sample is acquired. The cell's intensity output can be used to aid in the light-stripe system range calibration process. Scene reflectance information can also be derived from this intensity data.

One area of sensor design methodology being pursued lately involves moving circuitry out of the smart cells in order to pack the photoreceptive areas more tightly. This approach is motivated by the need to provide good optical efficiency in a practical sensor. Imaging optics must be designed to accurately illuminate the entire surface of the device. However, the gaps between photosites containing interleaved circuitry waste the light from the stripe that falls there. Moving circuitry from the photosensitive areas allows these gaps to be closed. The light collected by a given optical system is therefore used more effectively.

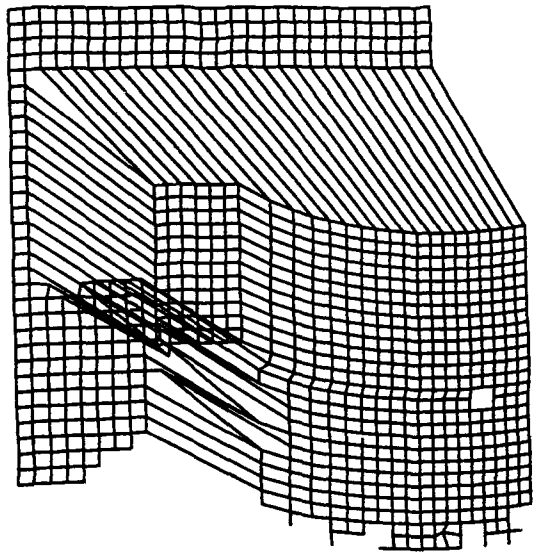


Figure 3: A 28x32 range image of a cup. This image can be captured within 1 to 2 msec.

Integrating sensing and processing on integrated circuit substrates holds great promise for developing of sophisticated data acquisition systems. Computation at the point of sensing allows raw sensor data to be tailored to the needs of higher level system requirements in a parallel fashion. The ability to acquire data intelligently also means that new sensing methodologies can be developed. One of the most distinguishing features of this research is that it is not just parallel implementation of known algorithms by VLSI technology to achieve increased speed, such as VLSI chips for convolution. Rather, we are demonstrating that integration of sensing and processing can make possible modifications of the operational principles of information acquisition (in our case, range imaging) which results in a qualitative improvement in performance.

5 Vision for Object Recognition and Manipulation

One of the most important applications for machine vision is robotic manipulation of objects. The CMU Image

Understanding group has been developing a method for modeling shapes, sensors, and tasks, and an automated method of constructing vision-guided object recognition and manipulation system by means of precompilation and learning.

5.1 Vision-Guided Sampling Using Deformable Surfaces

We (Hebert, Ikeuchi, Delingette) have developed a vision-guided manipulation system to collect small rock samples in natural terrain. Our initial system used superquadrics for shape representation and a simple clamshell gripper for manipulation. Recently, we have extended the system by using more accurate representations and finer control of the manipulation [16] so as to deal with complex scenes. We use free-form deformable surfaces as the base representation for the object models, implemented as a triangular mesh of typically five hundred nodes. The models are computed from surface features, range, surface normals, and curvature discontinuities.

For more dexterous manipulation, we constructed a three-finger gripper that allows for finer control of the grasping operations. Optimal grasping positions are computed by analyzing the local shape of the object at the three points of contact for every possible position of the gripper. The evaluation criterion is based on the sum of the areas of contact between finger and object at the three contact points. We have demonstrated the sampling task in complex scenes.

5.2 Vision Algorithm Compiler

The Vision Algorithm Compiler (VAC) [17] is a methodology that we have been developing over the years to provide an alternative to the traditional, expensive *hand-coding* of model-based vision systems. Given a vision task, a VAC starts with models of objects, sensors, and processing techniques. Working off-line, a VAC analyzes the models and automatically compiles a recognition strategy, which is compiled into an excecutable program which performs the task on-line.

During the past several years, work on the VAC has progressed in several areas. We have worked on modeling sensors [18], a frame-based geometric modeling system [19], generation of optimal classification strategies [20], accurate determination of object position and orientation [21], and an example system for bin-picking tasks [22].

Recently we (Ikeuchi, Kanade, and Sato) applied the VAC methodology to the task of recognizing a specular object in an ootical image and an object in SAR images, as well as determining its postion and orientation [23]. Figure 4 shows the overall system. The figure consists of sensor-independent modules (Aspect Generator, Template Generator, Aspect Classifier, and Fine Matcher), and a sensor-dependent module (Sensor Simulator). It

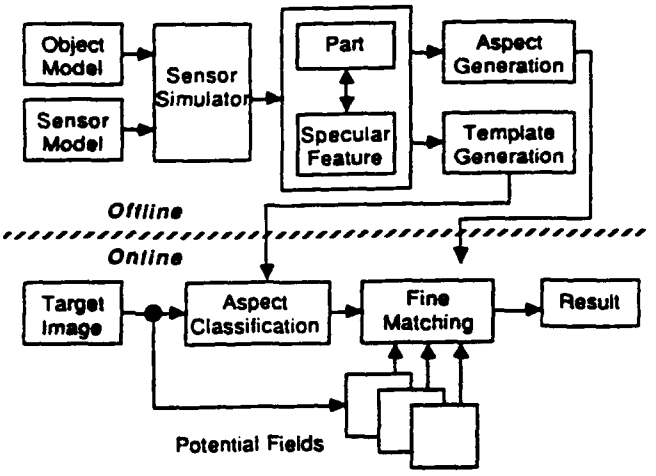


Figure 4: Precompilation method for object recognition

should be noted that most of the developed techniques are common between specular objects and SAR images except the sensor models.

In the off-line precompilation phase, images of the object are synthesized from a representative set of viewpoints that cover the entire viewing sphere. On the basis of the visible features, the sample images are grouped into aspects. Figure 5 shows the synthesized SAR images for an airplane. Each aspect is characterized by a matching template that models the appearance and distribution of the specular features for that aspect. For each aspect, a procedure is created and attached which should be used for calculating the precise pose of the object once an image is classified to belong to the aspect. In the on-line recognition phase, an unknown image is input to the system. According to the compiled strategy, the image goes through the two stages for recognition: aspect classification and fine matching. Figure 6 shows an example of the recognition result.

The system proved itself capable of localizing very specular objects in optical images. The only changes to the system required to deal with different image types was to change the sensor simulator model.

5.3 Assembly Plan from Observation

Another self-programming capability that we (Ikeuchi, Kang, Paul, Wheeler, Suehiro) have been working on are systems which "learn" or program themselves by observing a human performing the same task; we call this an *assembly-plan-from-observation* method. A human performs assembly operations in front of a TV camera of the system. The system recognizes such assembly operations

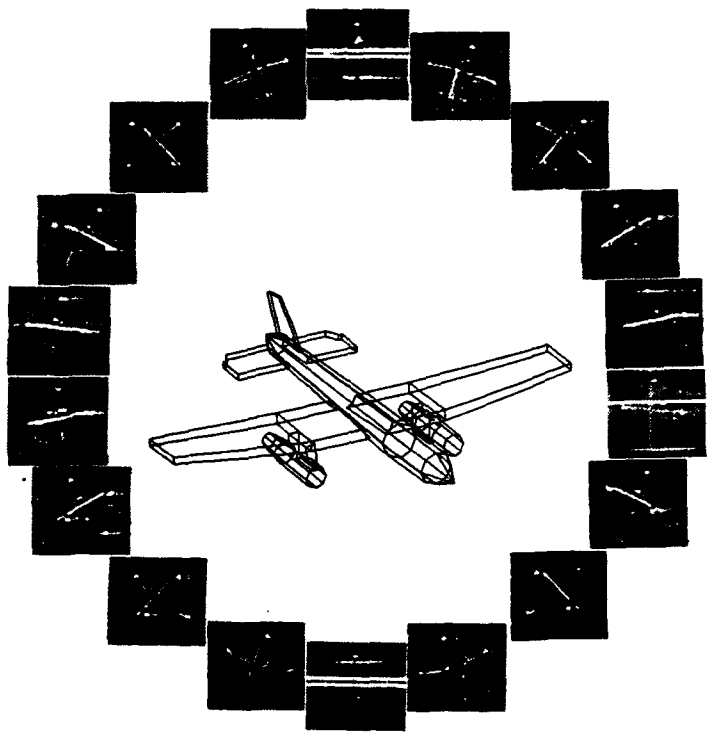


Figure 5: Simulated SAR images of an airplane

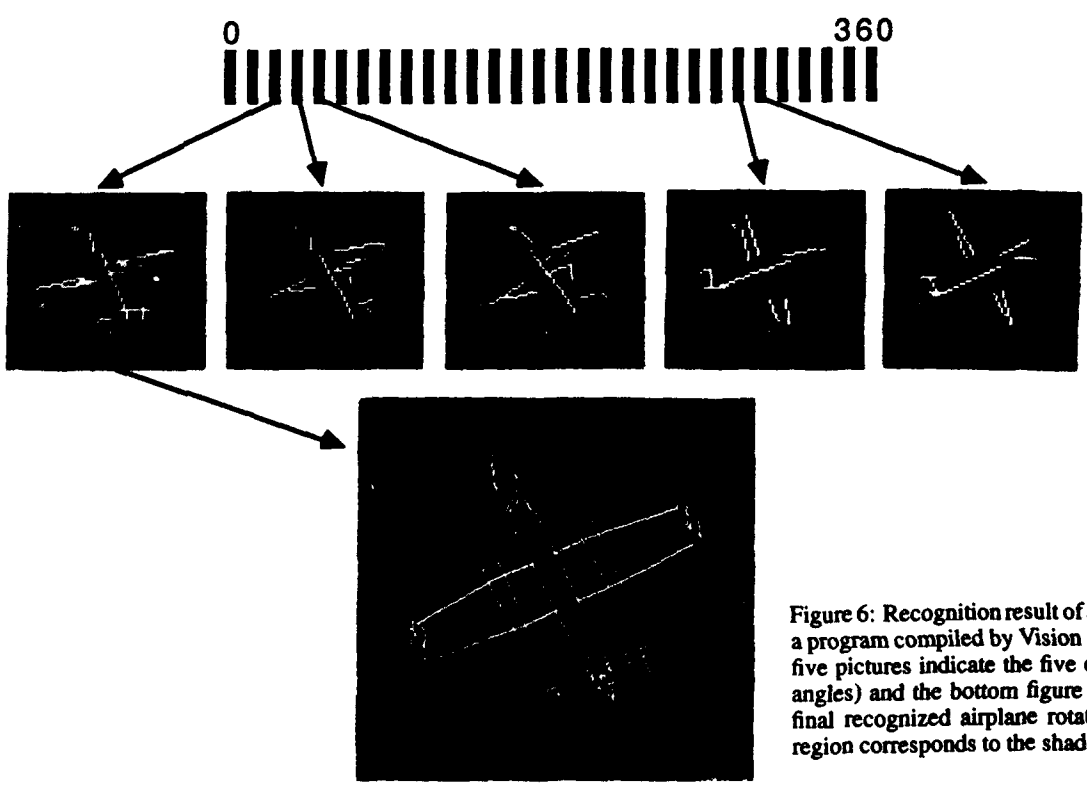


Figure 6: Recognition result of an airplane SAR image by a program compiled by Vision Algorithm Compiler. The five pictures indicate the five candidate aspect (rotation angles) and the bottom figure shows the overlay of the final recognized airplane rotation. Note that the dark region corresponds to the shadow of the airplane.

and generates an assembly plan to be able to repeat the same assembly operations using its robot arm.

This year, we have focused on a class of assembly operations, such as put-on or insert-into. One of the important purposes of such assembly operations is to achieve face contacts between objects. We have examined possible face contact relations among polyhedra and classified them into nine equivalent groups. We further examined the transitions which occur among these equivalent groups and the assembly operations which cause such transitions. The results are summarized into the *procedure tree* which associates transitions and necessary assembly operations among the groups.

The system works as follows. It recognizes object configurations at each step of assembly operations and then extracts face contact relations from these configurations. By consulting the procedure tree, the system can associate an extracted relation transition with the necessary operations to achieve the transition. Once a class of assembly operations is recovered from observation, by filling in necessary motion parameters, such as where to put or from which direction to insert, the system completes the learning of the assembly plan. By repeating this sequence, the system successfully generates the sequence of assembly operations necessary to repeat the assembly performed by the human [24]. Techniques to deal with noisy measurements have been also developed [25]

We plan to develop modules to recognize human grasping strategies and global path planning from observing a sequence of images.

6 Vision for Autonomous Mobile Robots

Mobile robots are vital for tasks in reconnaissance, exploration, and all missions to be carried out in remote locations. Visual navigation, however, is quite challenging because of the need for very robust sensing in changeable outdoor conditions and the complexity of situation assessment for a vehicle moving through the environment. CMU has several major programs in vision for mobile robot vehicles, including cross-country and roadway terrestrial navigation and underwater navigation.

6.1 Unmanned Ground Vehicle

Work on the Navlab [26], including perception, planning, and new vehicle construction, has now been grouped into a single project as part of DARPA's new UGV program. We (Thorpe and others) have a new vehicle, the Navlab II, which is a converted HMMWV. Perception algorithms running on the Navlab and Navlab II include road following, object recognition, and cross-country traversal. These vehicles navigate using using sonar, giga-hertz radar, and the ERIM laser rangefinder. We have recently demonstrated steering around obstacles and tracking guard rails and parked cars. Work on cross-country mobility has concentrated on achieving higher speeds through careful pipelining of the processing, and incorporating detailed models of vehicle control and kinematics. The best runs to date have achieved speeds of over 6 mph on moderate off-road terrain. EDDIE, the toolkit for building software architectures, and the Annotated Map data management system have been distributed to Caterpillar, University of Massachusetts, JPL, Florida Atlantic University, and other sites.

Pomerleau developed ALVINN, a neural net road following system [27], and has used it to drive the NAVLAB II up to 55 mph on highways, and for a continuous distance of over 21 miles. The networks are trained by watching a human drive, with synthetic noise added to the training examples to decrease sensitivity to features such as other vehicles and guard rails. ALVINN has also been modified to produce confidence measures in the output of the network[28]. This will allow several nets to be run in parallel, for instance nets trained on different types of road, and the output from the most confident net used for driving.

Road following using symbolic feature tracking in YARF (Kluge) now incorporates robust statistics and road models. YARF tracks features such as yellow center lines and white edge lines. Detected feature locations are recorded in a local map. After each image is processed, all the features in the vehicle's current vicinity are used to update the current model of road location, orientation, and curvature. Robust statistics approaches to weighting features are used to reduce sensitivity to outliers, and to give indications of possible intersections or changes in road structure. YARF routinely drives the Navlab along city streets near the CMU campus.

6.2 Planning for Robot Perception

Recent advances in mobile robots have demonstrated systems that can follow a simple lane or find a known landmark. The next step is to develop robots capable of driving in traffic, which requires interpreting much more complex road configurations, signs and signals, and interacting with other moving traffic. We (Reece and Shafer) are studying this problem, which we call tactical driving, with the Pharos traffic simulation program and the Ulysses computational driving program [Reece91a]. We began this work by developing Pharos, a traffic simulation program that models a network of streets. The input database describes the exact geometry of each lane of each street, and the details of each sign, signal and lane marking. Into this network of streets, Pharos injects vehicles we call "zombies" that turn randomly at each intersection, but follow actual driving laws and vehicle kinematics models as they travel. To make their driving decisions, ten times per (simulated) second, each vehicle follows a decision-making process by examining the data structures of Pharos to determine the lane of travel, con-

straints imposed by vehicles and traffic control devices, and intended maneuvers such as overtaking and turning.

To study robot driving, we developed a robot driving program called Ulysses that controls one vehicle in the simulated world of Pharos [Reece91a]. Whenever the "robot" vehicle is queried, a message is sent from Pharos to Ulysses to ask for the next steering and speed commands. Ulysses carries out a decision-making process that simulates how a (robot) driver would decide what to do. In this process, Ulysses sends "perceptual requests" to Pharos, such as asking for the upcoming lane geometry or the distance and speed of the vehicle in front of the robot. Pharos answers these requests just as a robot perception system would do. When it has run through its decision-making process, Ulysses reports its computed speed and steering commands to Pharos, which carries out that action and continues its simulation of traffic. The decision-making process of Ulysses is completely feasible for a robot vehicle.

Our chief finding to date concerns the perceptual cost of driving. Traditional AI planners are based on the notion that the perception system continuously provides a complete model of the world, within which the planner can find whatever it is interested in. However, the cost of finding all possible cars, signs, road markings, and other objects, ten times per second, while driving, is stupendous. It isn't even vaguely conceivable that a real robot driver would be able to provide such a continuously updated model of everything in the environment. Instead, the Ulysses program makes requests for the specific things it needs to know at each tick of the clock, in order to make the specific decisions it needs to make. In addition, since the perception requests are sequential, each one can constrain the geometry of the later requests. We carefully model the "perceptual cost" for Pharos to answer each request made by Ulysses, and have shown that this demand-driven model of perception is several orders of magnitude lower in cost than the complete world model assumed by traditional AI planners [Reece91b, Reece91c].

6.3 Underwater Perception

We (Hebert and Langer) have been working on underwater terrain modeling by sonar imaging for autonomous underwater vehicles (AUV). Recovering terrain shape from sonar data is complicated by the ambiguity inherent to sonar data, in which the effects of range and intensity are complicated by the shape, reflectivity, and roughness of the observed target, and by the transmission characteristics of sonar. Recovering shape from sonar data is therefore a highly underconstrained problem, much like shape recovery from visual data. Our approach is to use a reflection model that relates the observed range and intensity to all the unknown surface parameters. Using initial values of those parameters, shapes are reconstructed first in the neighborhood of high-intensity points which correspond to more reliable data. An iterative process is then used to revise both shape and surface parameters. The resulting sparse map is interpolated using a standard regularization-based interpolation algorithm.

We have tested the algorithm on a number of sonar images collected by Florida Atlantic University (FAU) [29, 30]. The results show that surface maps are correctly recovered even when very different types of terrain, such as rocks and sand, are present in the scene. This departs from previous algorithms which assumes implicitly that the bottom surface is homogeneous and flat. Our continuing research is in merging multiple sonar images and quantifying the performance evaluation of terrain reconstruction.

References

[1] Carlo Tomasi and Takeo Kanade. Shape and motion without depth. In *Proc. of the DARPA Image Understanding Workshop*, pages 258–270, Pittsburgh, Pa, September 1990.

[2] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams: a factorization method - 2. point features in 3d motion. Technical Report CMU-CS-91-105, School of Computer Science, Carnegie Mellon University, Jan 1991.

[3] Masatoshi Okutomi and Takeo Kanade. A multiple-baseline stereo. Technical Report CMU-CS-90-189, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 1990.

[4] I. S. Kweon and T. Kanade. High Resolution Terrain Map from Multiple Sensor Data. In *Proc. IEEE Intl. Workshop on Intelligent Robots and Systems*, pages 127–134, Tsuchiura, Japan, July 1990.

[5] M. Hebert. Building Object and Terrain Representation for an Autonomous Vehicle. In *Proc. Am. Control Conference*, Boston, June 1991.

[6] T. Kanade. Computer vision as a physical science. In Rick Rashid, editor, *Carnegie Mellon Computer Science:a 25-Year Commemorative*, ACM Press Anthology, pages 345–369. Addison Wesley, 1991.

[7] G.J. Klinker, S.A. Shafer, and T. Kanade. The measurement of highlights in color images. *International J. of Computer Vision*, 2(1):7–32, June 1988.

[8] G.J. Klinker, S.A. Shafer, and T. Kanade. A physical approach to color image understanding. *International J. of Computer Vision*, 4(1):7–38, Jan 1990.

[9] S.K. Nayar, K. Ikeuchi, and T. Kanade. Surface reflection: physical and geometrical perspectives. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7):661–634, July 1991.

[10] Katsushi Ikeuchi Shree Nayar and Takeo Kanade. Determining shape and reflectance of hybrid surfaces by photometric sampling. *IEEE Transactions on Robotics and Automation*, 6(4):418–430, August 1990.

[11] L. G. C. Hamey, Jon A. Webb, and I-C Wu. An architecture independent programming language for low-level vision. *Computer Graphics and Image Processing*, 48:246–264, 1989.

[12] Jon A. Webb. Adapt: Global image processing with the split and merge model. Technical Report CMU-CS-91-129, School of Computer Science, Carnegie Mellon University, April 1991.

[13] Jon A. Webb. Apply and adapt: Steps towards architecture independent image processing. *IEEE Computer*, to appear.

[14] Andy Gruss, L. Richard Carley, and Takeo Kanade. Integrated sensor and range-finding analog signal processor. *IEEE Journal of Solid-State Circuits*, 26(3):184–191, March 1991.

[15] T. Kanade, G. Gruss, and R. Carley. A very fast VLSI rangefinder. In *Proceedings of IEEE 91 International Conference on Robotics and Automation Conference*, pages 1322–1329. IEEE, April 1991.

[16] C. Francois, K. Ikeuchi, and M. Hebert. A Three-Finger Gripper for Manipulation in Unstructured Environments. In *Proc. IEEE International Conference on Robotics and Automation*, Sacramento, March 1991.

[17] K. Ikeuchi and T. Kanade. Towards automatic generation of object recognition program. *Proc. of IEEE*, 76(8):1016–1035, August 1988. (a slightly longer version is avaiable as CMU-CS-88-138).

[18] K. Ikeuchi and T. Kanade. Modeling sensors: Toward automatic generation of object recognition program. *Computer Vision, Graphics, and Image Processing*, 48(1):50–79, 1989.

[19] T. Kanade, P. Balakumar, J.C. Robert, R. Hoffman, and K. Ikeuchi. Overview of geometric/sensor modeler VANTAGE. In *Proc. the International Symposium and Exposition on Robots*, Sydney, Australia, November 1988. The Australian Robot Association.

[20] K.S. Hong, K. Ikeuchi, and K.D. Gremban. Minimum cost aspect classification: a module of a vision algorithm compiler. In *10th Intern. Conf. on Pattern Recognition*, Atlantic City, N.J., June 1990. (a slightly longer version is available as CMU-CS-90-124).

[21] K. Ikeuchi and K. S. Hong. Determining linear shape change: Toward automatic generation of object recognition programs. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 53(2), March 1991. a longer version, containing programs, is avaiable as CMU-CS-88-188.

[22] K. Ikeuchi. Generating an interpretation tree from a CAD model for 3-D object recognition in bin-picking tasks. *International Journal of Computer Vision*, 1(2):145–165, 1987.

[23] K. Sato, K. Ikeuchi, and T. Kanade. Model based recognition of specular objects using sensor models. In *Proc. of IEEE workshop on Directions in Automated CAD-based Vision*, pages 2–10, Maui, Hawaii, 1991. IEEE computer society.

[24] K. Ikeuchi and T Suehiro. Towards assembly plan from observation: Task recognition with polyhedral objects. Technical Report CMU-CS-91-167, School of Computer Science, Carnegie Mellon University, August 1991.

[25] T. Suehiro and K. Ikeuchi. Towards assembly plan from observation: Fine localization based on face contact constraints. Technical Report CMU-CS-91-168, School of Computer Science, Carnegie Mellon University, August 1991.

[26] Charles E. Thorpe. *Vision and Navigation: the The Carnegie Mellon Navlab*. Kluwer Academic Publishers, 1990.

[27] Dean A. Pomerleau. Neural network based autonomous navigation. In Charles E. Thorpe, editor, *Vision and Navigation: The Carnegie Mellon Navlab*, chapter 5. Kluwer Academic Publishers, 1990.

[28] J. Gowdy D. Pomerleau and C. Thorpe. Combining artificial neural networks and symbolic processing for autonomous robot guidance. *Journal of Engineering Applications of Artificial Intelligence*, to appear.

[29] D. Langer and M. Hebert. Building Elevation Maps from Underwater Sonar Data. In *Proc. International Symposium on Umanned Untethered Submersible Technology*, September 1991.

[30] D. Langer and M. Hebert. Building Qualitative Elevation Maps from Side Scan Sonar Data for Autonomous Underwater Navigation. In *Proc. IEEE International Conference on Robotics and Automation*, Sacramento, March 1991.

Progress in Computer Vision at the University of Massachusetts

Edward M. Riseman and Allen R. Hanson
Computer Vision Research Laboratory
Dept. of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

ABSTRACT[1]

This report summarizes progress in image understanding research at the University of Massachusetts over the past year. Many of the individual efforts discussed in this paper are further developed in other papers in this proceedings. The summary is organized into several areas:

1. Mobile Robot Navigation
2. Image Sequence Processing
3. Interpretation of Static Scenes
4. Image Understanding Architecture

The research program in computer vision at UMass has as one of its goals the integration of a diverse set of research efforts into a system that is ultimately intended to achieve real-time image interpretation in a variety of vision applications.

1. Robot Navigation

The UMass mobile robot navigation project continues to integrate a number of different algorithms with the goal of achieving robust landmark-based navigation. The primary component technologies are described in several sections of this paper. Sections 1.1 and 1.2 discuss the use of landmarks derived from a partial geometric model of the environment to determine the pose of the vehicle. Section 2.2 outlines one mechanism by which an initial partial model of the environment might be automatically acquired from a motion sequence. Section 3.1 discusses techniques for learning recognition strategies within the Schema object recognition system, which is capable of identifying naturally occurring objects. Our goal is to integrate natural objects into the landmark-based navigation system for outdoor navigation and to embed the results of this research into the planning and control framework developed by Fennema [18] which can effectively utilize landmarks at a number of levels, including low-level perceptual servoing for producing accurate motor actions, and plan-level perceptual servoing for maintaining adherence to a navigation plan.

1.1. Automated Model Acquisition and Extension

We are continuing our efforts towards the robust determination of pose (location and orientation) of the vehicle in a partially modelled 3D environment via the constraints derived from recognized 3D landmarks [25].

Recently, Kumar has performed experiments on model extension (Kumar and Hanson [25, 26] using basic techniques from pose determination. Points or lines whose 3D positions are known are tracked across frames using the line tracking algorithm of Williams [39], or the point-tracking algorithm of Sawhney [33]. From these points or lines, the relative orientation of pairs of frames are determined. The depths of unmodelled points, which are also tracked over the sequence, are then computed using triangulation. The sensitivity of the acquired depth to errors in the image center has also been investigated. In experiments using two image sequences for which ground truth is available, the 3D positions of the unmodelled points were recovered with an average error in depth of .25% and 1.3%. The error for the second case is larger than for the first in part due to the larger field of view (40^{o} compared to 22^{o}) which increases the sensitivity to errors in the location of the image center. Given that there must be some error in the original 3D positions of landmarks, recovery of new 3D points to this accuracy is a surprising and dramatic result.

1.2. Landmark Recognition

Beveridge [3] continues to develop his model-directed matching algorithms, which are being applied to landmark-based robot navigation. His previous research used a priori knowledge of an approximate

[1]This research has been supported in part by the Defense Advanced Research Projects Agency under TACOM contract number DAAE07-91-C-R035, HDL contract number DAAL02-91-K-0047, and ETL contract number DACA76-89-C-0016, by the National Science Foundation under grant CDA-8922572 and IRI-9113690, and by RADC under contract number F30602-91-C-0037

robot pose to project landmarks in a 3D model into the image plane. This allows 2D model lines to be matched with 2D data lines extracted from the sensory data by minimizing the error in the spatial alignment under rotation, translation, and scaling transformations in the plane. The resultant correspondence between landmarks and image features for the best 2D-to-2D match is used to recover the actual 3D pose of the robot. This system has been effectively applied to pose recovery in the UMass mobile robot project. However, this system may not be able to recover from the 2D distortion produced when the projection is done from an incorrect sensor pose.

A new 3D-to-2D model matching system has been developed to match 3D landmarks directly to 2D image features. During the iterative matching process, new 3D transformations between the world and the camera are computed, and landmark features are re-projected into the image. This accounts for perspective distortion during the search, and therefore allows recovery of the robot's true position more reliably than the 2D-to-2D matching system. However, the original system usually improves the initial erroneous pose estimate, and does so in roughly one fifth the time required by the second.

2. Image Sequence Processing

2.1. The p-Field: A Computational Model for Binocular Motion Processing

Balasubramanyam [1, 2] is developing an integrated framework for stereo and motion analysis. Given a binocular camera system moving through a static environment, it is possible to obtain a three-dimensional field of vectors, where each vector is parallel to the induced relative 3D motion of an imaged point and scaled in magnitude by the depth of that point. This 3D vector field, referred to as the p-field, is derived from optic flow and disparity, over a pair of stereo frames at successive time instants. The behavior of the p-field is examined for specific cases of restricted motion, as well as for general motion. In particular, the behavior of the p-field under translational vehicle motion promises to be more stable under small vehicle rotations than the behavior of the flow field. We expect that the p-field will allow more robust recovery of the sensor motion parameters, and tracking of 3D points through a sequence of images. Ultimately, this analysis should provide more robust recovery of 3D environmental information than independent stereo and motion analyses whose results are combined after the fact.

2.2. Reconstruction of Shallow Structures

In many man--made environments, obstacles in the path of a mobile robot can be characterized as *shallow*, that is, they have relatively small extent in depth compared to the distance from the camera. Sawhney [32] (these proceedings) presents a framework for segmenting shallow structures from the background over a sequence of images. Shallowness is first quantified in terms of *affine describability*. This is embedded in a tracking system within which hypothesized model structures undergo a cycle of prediction and model--matching. Structures emerge either as shallow or non--shallow based on their *affine trackability*. This work rejects continuity heuristics for purely image motion in favor of temporal continuity defined as the consistency of generic 3D models, namely shallow structures. This work will be applicable to obstacle avoidance and model acquisition by a mobile robot. In two indoor experiments, object structure represented as frontal planes was recovered to a depth accuracy in the range of 2-5%.

2.3. Multi-Frame Structure from Motion

Recovering structure from motion, even using information from multiple image frames, is difficult, partly because motion error can introduce large, correlated errors in the structure estimate. Thomas and Oliensis [35] (these proceedings) propose a method for recursively recovering structure from motion that can deal with this problem. The algorithm is based on the observation that errors in the motion produce cross-correlations in the structure errors across the 3D points. Conversely, these correlations are the record of the motion error. Thus, to explicitly incorporate motion error in a recursive algorithm, a record of the correlations in the structure errors must be maintained and updated.

Input for the algorithm consists of point correspondences tracked over many image frames. Horn's relative orientation algorithm [23] is used to provide two-frame structure estimates. For this algorithm, a somewhat complex error analysis is used to estimate the expected structure errors, including the cross-correlations. The fusing of the new structure estimate with the old is done using a standard Kalman filter, but with the cross-correlations taken into account. The results on synthetic images show that the structure estimates improve over time as expected; encouraging results on real images are also reported [30, 34, 35].

3. Interpretation of Static Scenes

3.1. Learning 3D Recognition Strategies

In an effort to automate aspects of model acquisition for image interpretation, Draper [13-16] has been examining the role of learning in model-based vision. In particular, he is addressing the automated construction of robust control strategies responsible for creating 'instance-of' relations during interpretation. Given a set of generic parameterized knowledge sources, the goal is to construct the Schema Learning System (SLS) which will learn (from a set of training images) a recognition strategy for a particular object class that minimizes the cost of recognition, subject to a set of accuracy constraints supplied by the user. Recognition strategies are represented by recognition graphs, which are similar in many ways to decision trees. Unlike decision trees, however, recognition graphs direct hypothesis creation as well as hypothesis verification. Object-specific strategies are learned in a two step process [15] (these proceedings). The first step involves learning which hypotheses should be generated. The second learns how to verify them efficiently. Thus, the task of SLS is to learn control and evidence combination strategies, not new models or knowledge sources. Initial experimental results demonstrate the potential of this approach.

3.2. View Description Networks

Model-directed object recognition becomes much more difficult when the viewpoint of the three-dimensional object is unknown [6-8]. Burns [5] (these proceedings) describes a system designed to effectively match a single 2D image of a potentially cluttered scene to a library containing multiple polyhedral objects and demonstrates its performance on several scenes. This approach to recognition can be characterized by three general ideas. *Description networks* optimize the search for matches to objects from a multiple object library by organizing information about the objects into a single network representation. *View descriptions* contain organized descriptions of the *projections* of the objects from views for which the objects are expected to be seen; these are used during the *match phase*. Finally, the correctness of view description matches are *verified* by estimating the 3D pose of the associated object, evaluating the estimation error and searching for additional assignments between object and images features, given the estimated pose.

An important part of this approach is the design of the *recognition phase* of the system: given a compiled view description network for an object library, the system must direct an effective search for the correct matches in cluttered images. This implies that three key problems are addressed: recognition from an unknown view, the indexing problem (selection of a few high probability candidates based on key features), and model based incremental recognition among the candidate competing hypotheses based on partial matches.

3.3. Model Extension Using Projective Invariants

Collins [9, 11] has developed a new approach to modeling man-made environments based on results from projective geometry. It is well known that the images of coplanar points and lines under rigid-body motion are related by a linear transformation in homogeneous coordinates. Given four known reference points or lines on the plane, the positions of all other points/lines on that plane can be reconstructed, regardless of camera position or intrinsic calibration parameters. Collins [10] (these proceedings) has extended these results; it is shown that it is possible to obtain partial and in some cases complete 3D reconstructions of those points and lines lying outside the reference plane. The main results are that with a calibrated camera, one reference plane tracked through two images is enough for complete reconstruction of the environment, while for an uncalibrated camera it is sufficient to have two reference planes tracked through two images. The effects of noise in the observations are considered, resulting in a general framework for data fusion in projective space.

3.4. Shape from Shading Revisited

Shape from shading has traditionally been considered an ill--posed problem. However, in recent work, Oliensis [28, 29] has demonstrated that the solutions to shape from shading are often well--determined, with little or no ambiguity. For the case of illumination that is symmetric around the viewing direction (i.e. the light source is behind the camera), it was shown in [27] that there is in general a *unique* solution to shape from shading. This proof is valid for *general* Lambertian objects (without holes), and is the first proof that the problem of shape from shading can be well--posed in general. These arguments were extended to the case of general illumination direction in [29], where it was demonstrated that, in this case also, the solutions to shape from shading are strongly constrained over much of the image.

Recently, Dupuis and Oliensis [17] (these proceedings) has developed a new approach to shape from shading, based on a connection with a calculus of variations/optimal control problem, and has demonstrated its performance on reasonably complex images. The approach leads naturally to an algorithm for shape reconstruction that is simple, fast, provably convergent, and, in many cases, probably convergent to the correct solution. The algorithm is robust against noise and, in contrast with standard variational algorithms, does not require regularization. An explicit representation is given for the surface: its height is expressed as the minimal cost for an optimally controlled trajectory.

4. The Image Understanding Architecture

The IUA project continues as a three-way collaboration between UMass, Amerinex Artificial Intelligence, Inc. (AAI), and the Hughes Research Labs (HRL); this coordinated effort is summarized in [36] in these proceedings. The first IUA prototype hardware has been assembled, tested, and is almost fully functional. The low-level processor for the second generation IUA and its controller have been designed, and a software simulator has been built for the controller and low-level array. The design for the intermediate level has just been completed.

UMass has developed a SIMD version of the Wormhole Routing technique that takes advantage of the Coterie Network in our low-level processor, in order to provide general permutation routing capability roughly equivalent in performance to that found in the Connection Machine, without the need for special hardware. The significance of this routing capability is that it allows us to build very compact, low-cost, mesh-based parallel processors, of reasonable size (up to about a million processors) that can perform general data-parallel processing.

Our experience with the Coterie Network has resulted in the description of a more general programming paradigm, called multi-associative processing. In turn, the capabilities of the Coterie Network have been explored for directly and indirectly supporting multi-associativity.

Consideration of a set of issues that must be addressed by a parallel symbolic database for intermediate-level processing is in progress. These include the problems of managing data from continuous streams of images, controlling persistence of the data, representations of the data, distribution of the data and maintenance of its consistency, and real-time systems issues.

A C++ class library has been implemented for an image plane data type that supports the development of low-level vision operations that are easily implemented by a parallel processor. This approach to parallel programming has the advantage that it does not involve a non-standard language -- it is merely a new object class written in C++. The only difference between a sequential implementation and a parallel implementation is the run-time library selected for linking.

5. Ongoing and New Work

5.1. Multi-Sensor Dextrous Manipulation

Grupen and Weiss [20, 21] are collaborating on a multi-sensor approach to dextrous manipulation in a robot workcell. Models of objects in the environment are constructed incrementally using an active sensing paradigm in order to support the ability to form stable grasp configurations with a Utah-MIT hand. The system consists of a camera mounted on one robot arm and the hand mounted on another. The transformation from the camera coordinate system to the hand coordinate system is computed using the pose refinement algorithm developed in [24].

One of the major issues with respect to modeling is fusing information from multiple views and different sensors. The particular application involves the integration of haptic and visual data to produce a triangulation of the surface of an object to be grasped. Haptic sensing here is the determination of the point of contact of the hand with the object based on force measurements. This gives a very rough estimate of the position and normal to the surface. The Giblin-Weiss [19] algorithm provides estimates of position, surface normal and curvature from a sequence of image with known camera motion.

5.2. Figural Completion from Principles of Perceptual Organization

Visual psychology provides strong evidence that generic knowledge of surfaces and occlusion is exploited very early in the perceptual grouping of image contours. Previous work by Williams [38] showed how generic knowledge of this sort could be captured as integer linear constraints and how the problem of segmenting simple scenes into (potentially overlapping) surfaces could be cast as an integer linear programming problem. The first system built along these lines demonstrated the completion of

gaps in straight sided figures, such as those caused by occlusion of one (opaque) surface by a second (opaque) surface, and subsequent recovery of the surfaces using only straight line interpolating contours. This limitation severely restricted the range of figures to which the system could be applied (reconstruction of occluded corners, for example, was impossible). In the past year, a new system has been built which uses cubic bezier splines of least energy as the interpolating contours. The system now captures curved illusory contours in figures that have been formulated by Kanizsa and other perceptual psychologists.

5.3. Perceptual Organization of Curves

Token-based grouping has thus far been applied to the problems of recovering straight-line structure [4] and more recently curvilinear structure [12] from the edge data of images. Dolan is currently extending this approach to local parallel implementations of this grouping paradigm. A SIMD model of curvilinear grouping has been designed to be implemented in the CAAPP layer of the IUA [37]. The model is relatively simple and promises many orders of magnitude speedup in extraction of straight and curved lines. A MIMD version is currently being designed which should alleviate the contention problems in the SIMD design by utilizing both the CAAPP and ICAP layers.

5.4. Qualitative Navigation

One way to solve the computational burden of maintaining accurate geometric maps for navigation is to eliminate such maps altogether. In contrast to model-based approaches to navigation where a map is required that explicitly represents the geometry and location of 3D objects in the world, Pinette [22, 31] is developing a method for qualitative, image-based navigation via homing. This approach maintains only a topological map of the world, representing particular places in the world and the directions between neighboring places. A place is represented explicitly in the map by the image of the world as seen from that location. Spatial reasoning is performed directly on images using only the bearings of landmarks from a current location and a neighboring target location, and does not need to acquire exact shape and range information. The work is developing a theoretical foundation for qualitative reasoning in the incremental homing paradigm, including cases with a lack of precision in recovering the direction of landmarks, and the presence of errors in landmark correspondence.

References

1. Balasubramanyam, P. and M. A. Snyder. (1991). "The P-Field: A Computational Model for Binocular Motion Processing," Proc. of IEEE CVPR, Maui, Hawaii, pp. 115-120.

2. Balasubramanyam, P. and M. A. Snyder. (1992). "A Computational Model for Binocular Motion Processing," Proc. of DARPA IUW, San Diego, CA.

3. Beveridge, J. R. and E. M. Riseman. (1992). "Can Too Much Perspective Spoil the View? A Case Study in 2D Affine and 3D Perspective Model Matching," Proc. of DARPA IUW, San Diego, CA.

4. Boldt, M., R. Weiss and E. Riseman. (1989). "Token-Based Extraction of Straight Lines," IEEE Trans. SMC, Vol. 19, pp. 1581-1594.

5. Burns, J. B. and E. M. Riseman. (1992). "Matching Complex Images to Multiple 3D Objects using View Description Networks," Proc. of DARPA IUW, San Diego, CA.

6. Burns, J. B., R. Weiss and E. M. Riseman. (1990). "View Variation of Point Set and Line Segment Features," Proc. of DARPA IUW, Pittsburgh, PA, pp. 650-659.

7. Burns, J. B., R. S. Weiss and E. M. Riseman. (1992). "Non-existence of General-Case View-Invariants" in Geometric Invariance in Machine Vision (J. Mundy and A. Zisserman, Ed.). To appear.

8. Burns, J. B., R. S. Weiss and E. M. Riseman. (1992). "View-variation of Point-set and Line-segment Features," IEEE PAMI. To appear.

9. Collins, R. and R. Weiss. (1990). "Vanishing Point Calculation as a Statistical Inference on the Unit Sphere," Proc. of IEEE ICCV, Osaka, Japan, pp. 400-403.

10. Collins, R. T. (1992). "Single Plane Model Extension using Projective Transformations," Proc. of DARPA IUW, San Diego, CA.

11. Collins, R. T. and R. Weiss. (1990). "Deriving Line and Surface Orientation by Statistical Methods," Proc. of DARPA IUW, Pittsburgh, PA, pp. 433-438.

12. Dolan, J. and R. Weiss. (1989). "Perceptual Grouping of Curved Lines," Proc. of DARPA IUW, Palo Alto, CA, pp. 1135-1145.

13. Draper, B. (1992). *Learning Object Recognition Strategies*. Ph.D. Thesis, Dept. of Computer Science, University of Massachusetts (Amherst). Forthcoming.

14. Draper, B. and A. R. Hanson. (1991). "An Example of Learning in Knowledge-Based Vision," Proc. of 7th SCIA, Aalborg, Denmark, pp. 189-201.

15. Draper, B., A. R. Hanson and E. M. Riseman. (1992). "The Schema Learning System," Proc. of DARPA IUW, San Diego, CA.

16. Draper, B. and E. M. Riseman. (1990). "Learning 3D Object Recognition Strategies," Proc. of IEEE ICCV, Osaka, Japan, pp. 320-324.

17. Dupuis, P. and J. Oliensis. (1992). "Direct Method for Reconstructing Shape from Shading," Proc. of DARPA IUW, San Diego, CA.

18. Fennema, C. (1991). *Interweaving Reason, Action, and Perception*. Ph.D. Thesis, Dept. of Computer Science, University of Massachusetts. Also COINS Technical Report TR91-56, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, September 1991.

19. Giblin, P. and R. Weiss. (1987). "Reconstruction of Surfaces from Profiles," Proc. of IEEE ICCV, London, pp. 136-144.

20. Grupen, R. and R. Weiss. (1991). "Force Domain Models for Multifingered Grasp Control," Proc. of IEEE Conf. Rob. & Aut., Sacramento, CA, pp. 418-423.

21. Grupen, R. and R. Weiss. (1991). "Issues in System Integration and Control for Interpreting and Manipulating the Environment," Proc. of IEEE Int'l. Sym. on Intelligent Control, pp. 221-226.

22. Hong, J., X. Tan, B. Pinette, R. Weiss and E. M. Riseman. (1990). "Image Based Navigation Using 360 Degree Views," Proc. of DARPA IUW, Pittsburgh, PA, pp. 782-791.

23. Horn, B. K. P. (1990). "Relative Orientation," IJCV, Vol. 4, pp. 59-78.

24. Kumar, R. (1992). *Model Independent Inference of 3D Information from a Sequence of 2D Images*. Ph.D. Thesis, Dept. of Computer Science, University of Massachusetts. Forthcoming.

25. Kumar, R. and A. R. Hanson. (1990). "Sensitivity of the Pose Refinement Problem to Accurate Estimation of Camera Parameters," Proc. of IEEE ICCV, Osaka, Japan, pp. 365-369.

26. Kumar, R. and A. R. Hanson. (1991). "The Application of 3D Pose Determination Techniques to 3D Model Extension," Proc. of IEEE Workshop on Passive Ranging, Princeton, NJ.

27. Oliensis, J. (1989). "Existence and Uniqueness in Shape from Shading," Dept. of Computer and Information Science, University of Massachusetts (Amherst), TR 89-109.

28. Oliensis, J. (1990). "Existence and Uniqueness in Shape from Shading," Proc. of IEEE ICPR, Atlantic City, NJ, pp. 341-345.

29. Oliensis, J. (1990). "Shape from Shading as a Partially Ill-Posed Problem," Dept. of Computer and Information Science, University of Massachusetts (Amherst), TR 90-50.

30. Oliensis, J. and J. I. Thomas. (1991). "Incorporating Motion Error in Multi-Frame Structure from Motion," Proc. of IEEE Workshop on Visual Motion, Princeton, NJ, pp. 8-13.

31. Pinette, B. (1991). "Qualitative Navigation," Proc. of IEEE Symp. IC, Arlington, VA.

32. Sawhney, H. and A. R. Hanson. (1992). "Tracking, Detection, and 3D Representation of Potential Obstacles using Affine Constraints," Proc. of DARPA IUW, San Diego, CA.

33. Sawhney, H., J. Oliensis and A. R. Hanson. (1990). "Image Description and 3D Interpretation from Image Trajectories," Proc. of IEEE ICCV, Osaka, Japan, pp. 494-498.

34. Thomas, J. I. and J. Oliensis. (1991). "Incorporating Motion Error in Multi-Frame Structure from Motion," Proc. of 7th SCIA, Aalborg, Denmark, pp. 950-957.

35. Thomas, J. I. and J. Oliensis. (1992). "Recursive Multi-Frame Structure from Motion Incorporating Motion Error," Proc. of DARPA IUW, San Diego, CA.

36. Weems, C., M. Herbordt, M. Scudder, J. Burrill and R. Lerner. (1992). "Current Status and Research in the Image Understanding Architecture Program," Proc. of DARPA IUW, San Diego, CA.

37. Weems, C., S. Levitan, A. Hanson, E. Riseman, J. Nash and D. Shu. (1989). "The Image Understanding Architecture," IJCV, Vol. 2(3), pp. 251-282.

38. Williams, L. R. (1990). "Perceptual Organization of Occluding Contours," Proc. of DARPA IUW, Pittsburgh, PA, pp. 639-649.

39. Williams, L. R. and A. R. Hanson. (1988). "Translating Optical Flow into Token Matches and Depth from Looming," Proc. of IEEE ICCV, Tarpon Springs, FL, pp. 441-448.

Progess in Image Understanding at MIT

W. E. L. Grimson, B. K. P. Horn, T. Poggio and staff
MIT AI Lab
Cambridge MA 02139

ABSTRACT

Our program in Image Understanding has continued to focus on the critical issues of selection, indexing, saliency computation and integration of visual cues. Efficient solutions to these problems are considered central to the development of robust object recognition systems, one of our primary research foci. We have also continued our work on the computation and the use of low level visual cues such as motion, stereo, color and texture, on analog VLSI circuits and on learning.

1 Introduction

Research at the MIT AI Lab has continued along a range of fronts, from low level processing, such as stereo, motion, color and texture analysis, through intermediate stages of integration of visual information, to higher level tasks such as object recognition and navigation. This report summarizes our main recent accomplishments in these areas. As usual in these reports, we refer interested readers to other publications for details.

2 Object Recognition

Because it has been one of our central focal points, we begin with our work in object recognition. In approaching the problem of recognising objects from noisy images of cluttered scenes, we have found it convenient to separate out three different aspects of the problem:

- **Selection:** Given a large set of image features, select (or group) subsets likely to have come from single objects.
- **Indexing:** Given one of these image feature subsets, select a small set of object models from the library that are likely to match the data.
- **Matching:** Given a data feature subset and an object model, determine if there is a legal transformation that would carry the model into a pose in the image that is consistent with the data, possibly by finding a matching between data and model features.

We will describe our recent work in each of these areas, beginning with the matching problem, since this must be solved even if we use trivial solutions to the other two problems.

3 Matching methods

The goal of recognition can be summarized as that of deducing the existence of a legal transformation from model to image and measuring the scope of the associated interpretation, i.e. is there an instance of the transformed object model in the scene, and how much of the model can be accounted for in the data. More formally, we define:

$$\begin{aligned} \{F_i | 1 \le i \le m\} &: \text{set of model features} \\ \{f_i | 1 \le i \le s\} &: \text{set of data features} \\ \mathcal{T} &: \text{legal transformation} \\ \ell : \{1, \ldots, s\} \mapsto \{1, \ldots, m, \star\} &: \text{a mapping.} \end{aligned}$$

Then we want to find the mapping ℓ_0, pairing data and model features (or excluding data features by pairing them with $\star$), from the set of mappings

$$\{\ell \mid \rho(\mathcal{T}_\ell F_{\ell(i)}, f_i) \le \epsilon \ \ \forall i \text{ such that } \ell(i) \ne \star\} \qquad (1)$$

that maximises

$$\ell_0 = \arg\max_\ell g\left(\{(i, \ell(i)) | i = 1, \ldots, s\}\right) \qquad (2)$$

where $\mathcal{T}_\ell$ is defined as the transformation associated with ℓ, ρ is some appropriate measure (e.g. Euclidean distance in the case of point features, or maximum Euclidean separation in the case of line features) and g is some measure of the magnitude of a match (e.g. number of matched model features, or total linear extent of such model features).

In recent years, we have considered three different classes of approach to this problem. The first concentrates on finding the correspondence ℓ, the second concentrates on finding the pose $\mathcal{T}$, and the third is a hybrid approach that combines aspects of both.

3.1 Interpretation Tree Search

The first class of methods searches for correct solutions in a **Correspondence space**, that is, a discrete s dimensional space, where each dimension correponds to one of the s sensory data features, and along each dimension of which there are $m + 1$ possible values, corresponding to the pairing of the data features to each of the m model features, or to the null character, indicating that the data features is extraneous. We have used the

idea of an Interpretation Tree [Grimson, 1990] to structure the search for correct correspondences in this space. The method searches a tree of all possible pairings of subsets of data features to model features, by executing a depth first walk along each of the dimensions of the space. This tree is in principle of exponential size, but effective use of constraints on the relative shapes of sets of features reduces the expected cost considerably. The method has been shown to be quite efficient at finding solutions in cluttered data, using either 2D data or 3D data. Recent formal analysis of the method by Grimson has led to the following results [Grimson, 1990]:

- If selection is perfect (no spurious data is included) and indexing is correct, then the expected amount of search is quadratic in the number of data and model features.
- If selection is not used and indexing is correct, then the expected amount of search is polynomial in the number of data and model features, but is exponential in the size of the correct solution.
- If selection is adequate (where a formal definition of adequate can be given as a function of the ratio of spurious data to model size), indexing is perfect, and one heuristically terminates the search once a sufficiently good solution is found (where formal methods for defining thresholds for "sufficiently good" are available), then the expected amount of search is quartic in the number of data and model features.
- If indexing returns an incorrect answer (i.e. one tries to match a model not present in the data), even using heuristic search termination, the expected amount of search to deduce that the model is not present in the data is exponential in the problem parameters.

Our conclusion from this is that good selection methods are essential for practical applications of Interpretation Tree methods. Although the results have been proven only in the context of constrained search methods like the Interpretation Tree, we suspect that the conclusions may well be relevant to other approaches to recognition. Because of this, much of our current research focuses on this problem, as detailed below.

3.1.1 Variations on Correspondence Search

One of the reasons that correspondence space search can have such a high cost is that it may waste time redundantly searching portions of the space that have already been considered. Thomas Breuel [1990] has developed a variation on this depth first search method that avoids this problem. In particular, as the interpretation tree is search, building up a correspondence, the method also concisely and efficiently keeps track of those regions of pose space that have already been explored by the search process. By doing this, the search process can be altered so that it does not re-examine geometrically equivalent interpretations of the data. This leads to an algorithm that has a worst case polynomial time performance, while preserving many of the practical advantages of correspondence space search.

3.2 Pose Space Search

The second class of recognition methods concentrates on finding the correct pose, rather than finding the correct correspondence. In the case of 2D objects constrained to a known plane, this in principle entails searching a 3D space, and in the more general case of 3D objects, this entails search a 6D space. Although this clearly would seem to be an improvement over the exponential approaches of correspondence space search, one generally requires a fine (or nearly infinitesimal) tesselation of the space, thereby increasing the cost of straightforward search methods.

In the ideal case of perfect sensor data, the tesselation of the pose space is generally not a problem. One can simply search over all possible pairings of model and image features, compute the associated transformation and vote for that transformation in pose space, a la Hough transforms. When uncertainty is allowed in the measurements, however, one must be more careful about voting for the entire volume of transformations consistent with the pairing of a noisy sensor measurement and a model feature, and this increases the demand on searching fine tesselations of the pose space.

One way around this problem is to exploit the geometry of pose space directly. Todd Cass [1990] has developed a powerful framework for doing this, by providing a formulation of the problem in which one can develop a polynomial-time algorithm that guarantees finding all feasible interpretations of the data, modulo uncertainty, in terms of the model. The approach is based on representing the model and the sensory data in terms of local geometric features such as vertices and line segments. It assumes bounds on the uncertainty in the position or orientation of the data features due to sensor error. One can show that there are only a polynomial number of quantitatively different transformations that align the model and the data modulo error. Object localisation is accomplished using a polynomial-time search through the set of all model transformations to find those that align large subsets of model and data features within the uncertainty bounds.

Intuitively, this approach can be considered as follows. For each pairing of a data and model feature, there is a set of transformations that will align the model features within the uncertainty region about the data feature. This set of transformations carves out a volume in pose space. If we consider all pairings of data and model features, we get a set of such volumes, and we are interested in finding points in the pose space contained within the intersection of a large number of such volumes. One could find such points by simply sampling points in pose space at some fine spacing, a method used earlier by Cass in implementing a very fast recognition scheme on the Connection Machine. It turns out, however, that one can efficiently find such volumes by decoupling the search over the full pose space into a coupled search over the translational components and a second search over the rotational components. Moreover, one can use the structure of these geometric arrangements to find very efficient, polynomial-time algorithms for finding the boundaries of these pose-space volumes.

A similar method has been developed by Breuel [1991].

3.3 Alignment Methods

The third class of methods combines aspects of the first two. In particular, the alignment [Huttenlocher & Ullman, 1987] method searches through portions of correspondence space to find possible matches of triples of image features to triples of model features. For each such pairing, the method then computes the associated transformation (which can be modeled as an affine transformation in the image) and uses this to transform the other model features into the image. These hypothesized features are then compared against the image data to verify or refute the correctness of the determined transformation.

3.3.1 Linear Combinations of Images

In its original form, the alignment method used a full 3D model of the known objects. Recently, Basri and Ullman [1990] have derived a new method for aligning objects with images that only requires a small number of images of the object as the model.

In particular, they show that a 3D object can be represented by a linear combination of 2d images of the object. If $\mathcal{M} = \{M_1, \ldots, M_k\}$ is the set of pictures representing a given object, and P is the 2D image of an object to be recognized, then P is considered an instance of $\mathcal{M}$ if

$$P = \sum_{i=1}^{k} \alpha_i M_i$$

for some constants α_i. If P is the image of a 3D object with sharp edges, then P can be represented as the linear combination of 3 other images of the object. By linear combination, we mean that the x coordinate of any point in P is simply a linear combination of the x coordinates of the corresponding point in each of the three canonical images. Similarly the y coordinate of any point in P is simply a linear combination of the y coordinates of the corresponding point in each of the three canonical images. While the two linear combinations for the x and y coordinates are different, the same two linear combinations hold for all points in the image.

As a consequence, if one can establish a correspondence between all of the points in the three canonical images, then an elegant solution to the recognition problem is to use the alignment method to establish an hypothesized correspondence between a small number (typically 3) of features in a new image and one of the canonical images, use this to compute the coefficients of the linear combination, apply that linear combination to the three canonical images to generate an hypothesized image of the model if the original correspondence is correct, and simply overlay that hypothesized edge image onto the original image and verify the hypothesis by comparing this edge image with the original image. Experiments with the method indicate that the hypothesized edge map can very closely overlap the actual edge map when a correct alignment correspondence is chosen.

While the basic method applies to objects with sharp edges, the same result holds for objects with smoth edges, in approximation. In particular, using earlier work by Basri and Ullman [1988], one can accurately predict the appearance of smooth objects, provided one has depth and curvature information about each contour point as well as the 2d contour image. Building on this idea, they show that one can extend the linear combinations method to smooth objects, where now any view of the object is represented as a linear combination of five canonical images of the object.

The method is initially formulated in terms of wire frame objects. When solid objects that can self-occlude are considered, one may need additional views to overcome the occlusion, but the same general approach holds.

3.3.2 Model Building for Alignment

Given the success of the basic linear combinations method, we have also developed several extensions to it. Since the heart of the modeling approach is to establish a correspondence between all the feature points in a set of canonical images, we need efficient methods for doing this automatically. One extension considers this problem within the more general context of matching contours in two images. This applies to a whole range of problems, including determining optical flow, matching features for alignment-based object recognition, and finding correspondence for long-range and apparent motion. Ivan Bachelder (together with Shimon Ullman) [1991] has recently finished the development of a scheme for matching partially constrained contours in two images using local affine transformations. It assumes that constraint lines, each narrowing down the match for a contour point in the first image to a line in the second image, are available for several contour points in the first image. The new scheme constrains the matching by assuming that contours are the orthographic projections of locally coplanar points, thereby reducing the recovery of correspondence to a local, linearly constrained, non-iterative calculation. Suggested applications include the determination of optical flow in short-range motion and the matching of aligned contour views in either alignment-based object recognition or long-range motion.

To determine the match for a contour point the scheme finds the best affine transformation, in a weighted least squares sense, that satisfies the match constraint lines and specially matched points within an oriented local neighborhood. This neighborhood is established by weighing the constraint equations. The weight for a given point constraint is the modulation of the Gaussian distance to the local origin, which establishes a circularly symmetric local neighborhood, by the Gaussian distance from the local tangent to the contour, which attempts to limit the neighborhood to a single contour. The width of the modulating Gaussian is set such that the axes of the oriented neighborhood are proportional to the local axes of inertia. To determine the width of the circularly symmetric Gaussian, the effective size of the local neighborhood, several sizes are simultaneously considered and the smallest one which yields a stable, unique solution, as determined by the condition number of a linear constraint matrix, is chosen. The maximum allowed condition number is set according to how much noise is expected in the constraint lines. In

the event that this condition number exceeds the maximum allowed condition number even at the largest neighborhood, there is more than one possible solution, from which we choose the smallest affine transformation which predicts matches for all neighborhood points that deviate the least from a purely translational matching. The neighborhood used to find the best pure translation for a point is again determined using the condition number criterion. Unique matches for are guaranteed through the use of a general pseudoinverse formulation that uses a modified singular value decomposition technique guaranteeing continuous solutions along contours. As the constraint matrix becomes less stable according to the condition number, the solution gradually changes.

This entire procedure is repeated for every contour point. Note, however, that the match for each point may be computed in parallel. Furthermore, since the predicted match for a point is simply given by the translational component of the local affine transformation, finding the match involves explicitly solving for only the two translational components of the affine transformation. A closed form solution for the match involves using a continuous version of the pseudoinverse to invert a 2×2 matrix and a 4×4 matrix, the coefficients of which are weighted summations of local point constraint parameters that may be determined in parallel.

Simulation results obtained on noisy synthetic and natural imagery have demonstrated the robustness of the approach and seem to agree with supporting biological evidence for the employment of such a scheme in the primate motion pathway.

Amnon Shashua has developed a second approach to the model building problem, specifically to the problem of recovering shape and correspondence from two orthographic views and 4 corresponding points. In attempting to address a problem traditionally seen in the context of three or more orthographic views, two arguments are raised: (i) shape should be represented as a deviation from a reference plane, rather than its traditional Euclidean form, and (ii) in the attempt to recover three-dimensional information from the changing 2D image, it may be more advantageous to address the problem of recovering shape and full correspondence/flow, rather than the problem of recovering shape and 3D motion parameters.

Shashua has shown that it is possible to recover shape and full correspondence/flow simultaneously, by using the instantaneous change in brightness as an integral part of the computational model. The resulting equations are further manipulated in way that the contribution of motion and shape to the displacement vector of a moving point are completely separated. The motion component is the transformation that aligns p, the image coordinates of an object point P, with $\tilde{p}'$, the image coordinates of the projection of P onto a reference plane in the second view. The remaining displacement between $\tilde{p}'$ and p', the corresponding point of p, depends only on shape. Shape is therefore defined as the deviation of P from a reference plane, defined by the 4 corresponding points, along the line of sight.

The separation of shape and motion is used to define a two-stage process in which motion is first factored out and then the unknown shape component is recovered using the instantaneous brightness change between both views. One advantage of this approach is that both stages are insensitive to the range of motion, thereby making it possible to deal directly with distant views. It is further shown that with this representation one can derive, in a straightforward manner, many previously established results relating to constraint lines, 3D motion parameters from two orthographic views and the prediction of novel views for three-dimensional object recognition.

Finally, the entire computational model requires simply to establish the affine transformation that aligns three corresponding points in both views with respect to the fourth corresponding point, serving as an origin, in both views. The affine parameters define a constraint line from which both shape and the location of the corresponding point are recovered by using the constant brightness equation as a second constraint line, and then finding their intersection.

An alternative exploration of building models by finding correspondences between sets of images has been developed by Ronen Basri. He has focused on the task of shape recovery from a motion sequence, which requires the establishment of correspondence between image points. The two processes, the matching process and the shape recovery one, are traditionally viewed as independent. Information obtained during the process of shape recovery, however, can be used to guide the matching process. Basri has developed a technique that builds on the constraints imposed on the correspondence by rigid transformations, extending them to objects that undergo general affine (non rigid) transformation (including stretch and shear), as well as to rigid objects with smooth surfaces. In all these cases corresponding points lie along epipolar lines, and these lines can be recovered from a small set of corresponding points. Basri has developed an algorithm that takes advantage of such epipolar lines to recover the correspondence from three contour images. The algorithm has been implemented and used to construct object models for recognition.

3.3.3 Constraints on Model Images

On a related note, the original linear combinations approach used as its model a set of edge maps taken under similar conditions in which the points on the edge maps had been placed in correspondence. The study of Mooney images (high contrast images of faces) shows that the setting of light sources plays a significant role on the interpretation of the image of a 3D object. Amnon Shashua has conducted a computational study showing that lighting conditions can be compensated for prior to matching the image of the object with its internal model. The entire computation can be carried out by a single layer *perceptron*. The study presents three observations: (i) an image of an object taken with an arbitrary lighting condition can be expressed as a linear combination of three images, each taken from a different direction of light source, (ii) the linear coefficients can be recovered from the zero-crossings, provided their location is

accurate, and (iii) accuracy can be traded off by using the sign-bits instead of zero-crossings for recovering the linear coefficients.

3.3.4 Correspondence for Alignment

Independent of how the object models are built, alignment methods still need to establish a correspondence between the image data and the model. The usual method used for determining correspondence is to compute features explicitly from images and then match them to their equivalents in models. This approach may be problematic, however. First, it is difficult to compute the features reliably. Second, the possible pairings of image and model features is large and the search for a solution may be extensive. In the past year, Shimon Ullman and Pam Lipson have been investigating schemes to solve the correspondence problem, and therefore determine alignment, without explicitly computing features. They have explored three techniques based on Ullman and Basri's linear combination approach for object recognition. The goal is to find coefficients that align a model with an image. They start with a rough alignment stage. They then improve the correspondence by performing a simultaneous matching of points. The techniques are designed to handle models and images separated by relatively small rotations of up to 30 degrees. They have tested the three techniques on synthetic and natural objects. The results to date are promising. They are currently evaluating the performance of the techniques as the output from the rough alignment stage degrades.

3.4 Alternative matching methods

An alternative formulation of the matching problem has been explored recently by Sandy Wells [1990], focusing on finding statistical characterizations of feature based recognition, attempting to balance simplicity with accuracy. The method uses a maximum a-posteriori probability (MAP) criteria within the general context of model-based recognition.

Wells has derived a simple MAP model matching criterion that captures important aspects of recognition in controlled situations. A detailed metrical object model is assumed. A probabilistic model of image features is combined with a simple prior on both the pose and the feature interpretations to yield a mixed objective function. The parameters that appear in the probabilistic models can be derived from images in the application domain. By extremising the objective function, an optimal matching between model and image features results. Within this framework, good models of feature uncertainty allow for robustness despite inaccuracy in feature detection. In addition, the relative likelihood of features arising from either the object or the background can be evaluated in a rational way. The objective function provides a simple and uniform means of evaluating match and pose hypotheses by the amount of the image that is explained in terms of the model, as well as the metrical consistency of the hypothesis. It allows these two aspects to be traded off in a rational way based on domain statistics. An experimental implementation of MAP model matching, among features derived from low resolution edge images, has been built and tested.

4 Selection methods

For all of the matching systems we have developed for recognition, performance both in terms of speed and accuracy depends critically on the set of data features supplied to the algorithm. Thus, all of these methods can benefit from good selection methods, that isolate subsets of the data features likely to correspond to a single object. While model-driven selection methods, such as the Hough transform, or Geometric Hashing are of use with small libraries of objects, we are interested in methods that also work with large libraries, and this suggests the use of data-driven selection methods.

4.1 Saliency

One method for selecting groups of data features on which to focus is the saliency networks developed by Shashua and Ullman [1988] , the basis of which was described in earlier proceedings. The key idea is to use local measures of curvature and orientation as part of a global optimization process to find contours that appear most salient relative to the entire set of image contours.

Shashua and Ullman [1991] have continued to extend this work, by developing a network model to perform grouping of image contours. The input to the net are fragments of image contours, and the output is the partitioning of the fragments into groups, together with a saliency measure for each group. The grouping is based on a measure of overall length and curvature. The network decomposes the overall optimization problem into independent optimal pairing problems performed at each node. The resulting computation maps into a uniform locally connected network of simple computing elements.

A grouping of the image contours is defined as the formation of a set of disjoint groups, each corresponding to a curve that may have any number of gaps, and whose union covers all the contour fragments in the image. Given a function $F(A)$ that measures some desired property of a group A, we would like to find a disjoint set of groups $A_1, ..., A_m$ that maximizes $\sum_i F(A_i)$ over all possible groupings. The optimization problem is defined more specifically below.

For the purpose of grouping it is convenient to consider the image as a graph of edge elements. The vertices of the graph correspond to image pixels, and the arcs to elementary edge fragments. The input to the grouping problem is a contour image, represented by a subset E' of the elements in the graph. A path in the graph corresponds to a countour in the image having any number of gaps.

A grouping of these elements is a collections of chains of elements $A_1, ..., A_m$ such that $A_i \cap A_j = \emptyset \;\; i \neq j$ and $\cup_i A_i \supseteq E'$. To define an optimal grouping we will define a function $F(A)$ that measures the quality of a group A. An optimal grouping is then a grouping that maximizes $\sum_{i=1}^{m} F(A_i)$ over all possible groupings of the elements.

This problem is similar to the saliency computation and it is motiviated by similar considerations of finding smooth long boundaries. The difference is that now

we are trying to form explicit grouping, and the optimaility measure is therefore not just evaluated at each image location independently, but is defined instead over possible groupings. This precludes, for example, a part of a contour from participating in a number of different groups.

4.2 Visual Attention

A second way to build on the saliency concept is to encorporate the idea of visual attention. Tanveer Syeda [1990, 1991] has been developing a computational model of visual attention, and has used it to demonstrate attentional selection as a useful module for object recognition.

Towards this end, two modes of human attentional behavior, namely *attracted attention* and *pay-attention* modes were identified. The attracted attention mode of behavior is spontaneous and is commonly exhibited by an unbiased observer (i.,e. with no a priori intentions) when some object or some aspect of the scene attracts his/her attention, while the latter is a more deliberate behavior exhibited by an observer looking at a scene with *a priori* goals (such as the task of recognising an object, say) and hence paying attention to only those objects/aspects of a scene that are relevant to the goal.

Briefly, the model suggests that the scene represented by the image be processed by a set of interacting feature detectors that generate a hierarchy of maps, representing features such as brightness, color, texture, depth, grouping of edges, and others such as shape, size, symmetry, etc. The feature maps are then processed by filters incorporating strategies for selecting distinctive regions in these maps. The choice of these strategies is guided by a central control mechanism that combines top-down task level and *a priori* information with the bottom-up information derived from the features, to demonstrate either mode of attentional behavior as desired. Finally, an arbiter module housing another set of strategies selects the most significant features across the feature maps, which can then be used in, say an object recognition system.

A system implementing the computational model described here is being built. The aim of this is to demonstrate how selection can be achieved via attention and specifically, how it can be of use in object recognition. So far, three features were chosen, namely, color, texture, and parallel-line-groups. The respective feature maps were built, and the selection filters for finding distinctive regions in these maps have been developed. In addition, a version of the arbiter module to combine the saliency information from the various features has been built. Some of this work is now described.

The color feature map was developed that describes the color image as consisting of perceptually different colored regions. Here, a method of perceptual categorisation of a color-space was introduced that made possible fast color region segmentation. A color saliency map was then built which used a color saliency measure that emphasised attributes that are also salient in human color perception.

The texture feature map was generated by regarding the image as being generated by a space-limited stationary stochastic process. Here, the segmentation of the textured image was obtained by a comparison of the linear prediction spectra of adjacent windowed regions of the image. Properties such as the relative distribution of dark and bright blobs were then made use of to judge the distinctiveness of a region. This was used to generate the texture saliency map.

Lastly, the parallel-line-groups feature map highlighted groups of closely-spaced parallel lines in an edge image (i.e., brightness image passed through an edge detector). It has been found that some texture information can be modeled this way. For example, printed letters on a surface (such as a bottle) appear as a bunch of closely spaced parallel lines when passed through an edge detector. Similarly, some types of wooden tables show this type of texture in an image.

The three feature-maps, the respective saliency-maps, and the arbiter module built constituted the implementation of the attracted-attention mode of the model. The next phase of the work implemented the pay-attention mode. This was built with the aim of performing selection in model-based object recognition. Here, the color and texture information in the model (extracted using the feature maps described earlier) was used to build a description of the object-model. This description was then used to design strategies for the selection filters. This involved developing new algorithms for finding instances of regions in the image satisfying object-model color and texture descriptions.

Finally, a 3D from 2D recognition system was built to evaluate the selection mechanism. Initial studies with selection based on color information alone, have been encouraging. Much of the future work will concentrate on the integration of the selection mechanism with the recognition system, and also developing faster ways of matching model features to data features that exploit information derived from the selection process.

4.2.1 Other uses of color

Another approach to the selection problem has been developed by Kah Kay Sung. Similar to Syeda, his work has focused on using color as a cue for finding salient regions. Color is an excellent visual cue for gathering surface material information from the scene, because if treated appropriately, it factors out image effects due to surface orientation changes and illumination differences. The specific approach involves extending traditional signal and image processing operations to work with color data. The ultimate goal is to establish a formal relationship between early level scalar field image processes and their corresponding vector field notions, so that color images can be treated and operated upon as piecewise continuous fields like grey-level intensity images.

There are three main aspects to this work. First, a color representation scheme and a related color difference measure are derived that respond strongly to actual hue differences in an image but not to pure intensity changes. Both notions are based on a pigmentation model of material surfaces, where the Lambertian color of a surface depends on the composition of its embedded pigments. Second, by building on an earlier radiometric technique by Wolff at Columbia, an extended method for extract-

ing Lambertian color components has been developed. One can use this technique to produce reduced specularity images. Finally, Sung has extended scalar field image processes to color, by deriving color analogues for some early level scalar image processes and concepts, based on his chosen color representation scheme. The list includes: (1) a model and quantitative measure for color noise, (2) noise reduction techniques, for example smoothing, MRF like local averaging, and median window filtering, (3) boundary detection, and (4) uniformity detection as an approach to region growing. All the above mentioned color image processes have been implemented and tested either on the parallel Connection Machine or on the Symbolics Lisp machines.

Working jointly with Brian Subirana, Sung has used the color based approach to region growing to tackle the problem of perceptual organization, or selection, of salient image regions. Using the technique of Curved Inertia Frames, developed earlier by Subirana-Vilanova, they have devised a perceptual organisation scheme that works without the need of explicit edges. Their approach is based on the concept of finding brightness ridges in images, and utilises a new ridge detector which is independent of scale and which can be applied to texture, color and brightness. They have implemented the scheme for perceptual organization using color on the Connection Machine.

4.3 Region Based Grouping

An alternative approach to the selection problem was explored by David Clemens [1991]. He has developed a region-based grouping method that attempts to collect together image edges likely to have come from a single object. To do this, Clemens' method uses a variation on a brushfire technique to propagate labels from edges into regions of uniform intensity. Once the propagation terminates in a set of region centers, a backtracking process is initiated that traces the lables back from the region center to identify those edges that "surround" it. The process careful to allow for a branching of the boundaries between regions of constant label in this backtracking stage, thereby avoiding some of the standard problems associated with brushfire methods. This method finds sets of edges that bound a region of roughly uniform intensity. Because of this, one can use the center of the region to establish an origin relative to which an ordering can be established on the collection of edges. A similar ordering can be imposed on model edges, and this ordering can be used in the matching process to dramatically reduce the cost of the search.

5 Indexing and invariants

Indexing remains a major problem for recognition systems. Since the cost of incorrectly indexing an object model is high (based on our earlier complexity results), we seek indexing methods that can avoid the obvious linear search method. In previous proceedings, we have reported on some of our work on this problem [Breuel, 1989]. In the past year, we have continued to explore other approaches to this problem, with particular focus on the use of invariants, the use of non-accidental properties, and indexing treated as the problem of matching models to images by storing markers for the models in a hash table at compile time, which the images access at run time.

Invariant functions have been widely used for the recognition of planar objects from arbitrary views. They form an attractive option, since in principle one can simply compute an invariant from the image and directly look up the (unique) object that could produce that value in a precomputed lookup table. Questions of the stability of such schemes are unclear, however. Furthermore, Clemens and Jacobs [1991] have recently shown that there are no non-trivial invariant functions when models consist of arbitrary collections of 3D points, and projection is modeled as orthographic with scale (a similar result has been shown independently by Burns, et al., [1990] and by Moses and Ullman [1991]). Jacobs (in this proceedings) considers in what restricted 3D domains invariant functions might be possible. That paper shows, for example, that there are invariant functions that do not produce false positives matches only when the set of allowable models is a measure 0 subset of the set of all models containing 3D point features. Moses and Ullman explore consider the effect of restricting recognition to certain classes of objects, but still show that for some classes there are no non-trivial invariant functions. Despite the negative tone of these results, we are continuing to explore the role of invariants in indexing and recognition.

A related approach to indexing is to consider creating the simplest and most economical description possible of the set of all images that an object may produce. David Jacobs has developed solutions for this problem in a variety of instances, including models that consist of point features, models that consist of point features with rotational degrees of freedom, and models that consist of points with tangent or directional information. These results have led to new methods of implementing indexing, as well as some new insights into approaches that use invariants and non-accidental properties. In addition, Jacobs has developed a partial solution to the problem of accounting for sensing error when implementing alignment, indexing or invariant-based approaches to object recognition with models that consist of point features.

6 Error analysis

Besides building and testing implementations of recognition systems on real data, we have also devoted some of our effort to considering formal analysis of the behavior of recognition systems, especially in the presence of bounded uncertainty in the image measurements.

Eric Grimson, in collaboration with Dan Huttenlocher (Cornell), has continued to develop a framework for such error analysis, based on the following model. Assume that an object is modeled by a set of simple geometric features, such as points or lines. Assume that similar geometric features are extracted from the image, but that their position is known only to within some bounded error, e.g. a point's position is known only to lie within a disc of radius ϵ. Under these conditions, we are in-

terested in analysing the performance of a recognition system as the sensor uncertainty changes, and as the amount of scene clutter changes. We are typically interested in the probability of a false positive result (the system incorrectly identifies an instance of the object in the scene), and the conditions needed to guarantee that this probability is less than some bound.

In earlier work, we developed an occupancy model of this problem, and applied it to an analysis of the Hough transform [Grimson and Huttenlocher, 1990]. We derived conditions under which the Hough transform could be guaranteed (within the assumptions of the analysis) to provide no false positives, as well as showing how the method should degrade with increasing scene clutter (an observation that has been experimentally independently verified by other groups). In this work, we considered the recognition of 2D objects in a known suppor plane from single images, and 3D objects from 3D data.

More recently, we have extended our analysis to consider the case of recognizing 2D and 3D objects from a single 2D image, using the weak perspective projection model [Grimson et al. 1991]. For the case of 2D objects, the effect of the projection can be modeled as an affine transformation of the plane. Based on this, a number of recognition systems use the idea of choosing three points as a basis for an affine coordinate system, and rewritting the coordinates of all other points with respect to this system. Such affine coordinates are invariant under any affine transformation, and can in principle be used to drive the recognition process. Together with David Jacobs, we have shown that under the bounded error model, the range of values for the affine coordinates of a point, written with respect to some basis triple of points, is given by an elliptical region in affine space, where the size, orientation and position of the region depend on the actual values of the points used to establish the affine basis. Because of this, we have shown that methods, such as Geometric Hashing, which rely on building a lookup table of invariant coordinates at compile time can no longer do so, since the region of consistent lookup values associated with a point depends on information that is accesible only at run time. At the same time, we have shown that the range of positions of a fourth point, defined by some affine coordinates relative to an affine basis and where there is bounded uncertainty in the positions of all of the points, is simply a disc whose size depends only on the affine coordinates, and not on the actual values of the basis points. This means that methods such as Alignment, which project points back into the image for future verification, do not suffer from the problem mentioned above. As well, this analysis allows us to precisely define the regions over which a verification system should look for supporting evidence of an object. Using this, we have been able to analysis the false positive rates for both Geometric Hashing and Alignment, showing how both will degrade with increasing noise and with increasing clutter, but also showing how, as originally-defined, the alignment method tends to perform better.

Recently, together with Tao Alter, we have extended this analysis to the case of an arbitrary 3D object, with similar conclusions. The analysis derives approximate expressions for the range of values of the parameters of a 3D transformation that are consistent with the pairing of triples of model points with triples of uncertain image points. While these expressions give us a sense of the uncertainty associated with individual parameters of the transformation, and allows us to predict overestimates for the range of possible positions associated with a project model point, there is still room for tighter bounds. Along these lines, Alter has been developing a second scheme for alignment-based verification of rigid objects in the presence of error. He uses a Monte Carlo sampling method to empirically derive the shapes and sizes of the propagated uncertainty regions when there is a "least-commitment" set of three correspondences, and has been extending this to examine how the regions shrink when more correspondences are added, and, in particular, how many additional matches are needed before the regions are expected to vanish. This results of this analysis should enable us to devise methods for conducting a probabilistic search for a complete set of correspondences, so as to more efficiently reach the goal.

In a different vein, Ronen Basri has examined the error characteristics of recognition of smooth objects. The recognition of objects with smooth bounding surfaces from their contour images is considerably more complicated than that of objects with sharp edges, since in the former case the set of object points that generates the silhouette contours changes from one view to another. The "curvature method", developed by Basri & Ullman [1988], provides a method to approximate the appearance of such objects from different viewpoints. Basri has derived an error analysis of the curvature method. He applies the method to ellipsoid objects and computes analytically the error obtained for different rotations of the objects. The error depends on the exact shape of the ellipsoid (namely, the relative lengths of its axes), and it increases as the ellipsoid becomes "deep" (elongated in the Z-direction). He shows that the errors are usually small, and that, in general, a small number of models is required to predict the appearance of an ellipsoid from all possible views. Finally, he has shown experimentally that the curvature method applies as well to objects with hyperbolic surface patches.

7 Low level processing

While much of our work has focused on recognition, we have also continued investigations into other aspects of visual processing, most notably low level measurement processes that supply the data on which recognition and navigation methods apply.

7.1 Motion

Work on exploiting the fixation of image sequences is continuing. Ali Taalebinezhaad [1990, 1991] has developed a simple method for creating a synthetic fixated sequence from a 'normal' (unfixated) image sequence. The result can be used by his direct motion vision algorithm for recovering translation and rotation of the camera, without requiring the explicit computation of either *feature correspondence* or *optical flow*. Work is currently

underway to test an implementation of the method on real image data.

Non-linear optimisation methods for recovering camera motion are being explored by David Michael. They yield very good results from three-image sequences. Unfortunately they are computationally very expensive. So they mostly serve at the moment as a way of getting 'ground' truth about actual camera motion. David is also working on better methods for obtaining calibrated image sequences. This is useful since it is very difficult to get accurate information by mechanical means (i.e. accurate enough measurements of camera translation and rotation).

Several algorithms for finding the focus of expansion (FOE) - the point to wards which the camera is moving - have been simulated and compared by Ignacio S. McQuirk [1991] on synthetic and real image sequences. He was able to make a selection of a good least squares direct motion vision algorithm. This is now being implemented in analog VLSI in research sponsored under another contract (from NSF and DARPA jointly).

7.2 Calibration

Together with John Wyatt, Berthold Horn has developed a new 'camera calibration' method (interior orientation). Before a camera can be used for metric purposes, the principal distance (a.k.a effective focal length) and the principal point (where the perpendicular from the projection center pierces the image plane) must be known. Present methods for calibrating cameras require expensive fixtures, granite tables and temperature controlled rooms. The new method, implemented by Jesus Domingues [1991], depends on the fact that the image of a sphere under perspective projection is an ellipse with major axis passing through the principal point, and with eccentricity a function of the ratio of the distance of the ellipse from the principle point to the principal distance. A least squares procedures obtains good estimates of interior orientation from a large number of images of black balloons against a bright background. Such a method makes it possible to calibrate cameras in an industrial setting where the complex equipment used by photogrammetrists is not available. It also does not require a carefully machines calibration object. It may prove useful in mobile robotics, where cameras may be jarred and need to be recalibrated from time to time.

7.3 Lines, Edges and Regions

David Mercer [1991] has implemented and improved a line-finding method (as opposed to edge finding). 'Lines' are ridges and valleys in the brightness surface. They often carry information complementary to that found in edges, and also carry useful information in the case of images that do not have any real edges. The method depends on finding the sero-crossings of the dot-product of the gradient vector and the direction of most negative (or most positive) second directional derivative. To be useful, the output has to be filtered by considering both the magnitude of the gradient and the magnitude of the Laplacian. Line-drawings - as opposed to edge drawings - carry useful information in the case of images of natural objects - as opposed to polyhedral man-made objects. They provide for tremendous data reduction and hence band-width reduction in video transmission. The result is presented on a grey background, with bright lines in white and dark lines in black. Interestingly, an approximation of the original image can be reconstructed from an a line drawing combined with an edge drawing. The result is nowhere near correct photometrically, but looks very similar to the original image.

In a related development, Lisa Dron has implemented a new segmentation algorithm that allows matching of images even when they are taken under quite different lighting or climatic conditions. One feature of the algorithm is that the boundaries between regions are found accurately -not displaced by prior smoothing employed by some alternate algorithms. One possible application of the new algorithm is in image compression, since an image can be reconstructed from the segmented image along with some auxiliary information using, for example, a resistive grid. The result is more accurate than the method described above - i.e. the grey-level are actually close to those of the original image - but requires that more information be transmitted. Efforts are also underway to implement this algorithm in an analog VLSI test circuit.

7.4 Contour Texture

An oak leaf can be easily distinguished from an elm leaf, yet the same oak leaf cannot be distinguished from other oak leafs unless a detailed inspection is performed. Brian Subirana has been examining the idea that complex region boundaries are processed by the visual system so as to obtain a simple abstract description, called Contour Texture, which contains much less information than that provided by the location of all the points of the contour. The frequency of protrusions (regardless of their shape), the dominant orientation or the average curvature along the curve are examples of contour texture properties. Contour texture may be useful in several domains, including recognition, indexing and grouping. While Contour Texture clearly differs from traditional image texture, the current scheme is based on existing filter-based approaches to texture (e.g. [Malik and Perona 1990]).

7.5 Invariants

Berthold Horn has begun to investigate "invariants" that apply in the case of perspective projection - as opposed to colineation. The idea is, of course, to be able to recognize object without explicitely recovering the attitude of the object, or as a way of filtering search in recognition and attitude finding. An example of this may be found in the Lisp book (Winston & Horn) where such methods are used for matching images of star fields. While allowing arbitrary linear transformations leads to areas of mathematics that have been explored before, it gives up a lot of constraint over an approach that, while less tractable mathematically, models the actual image formation process more accurately. An invariant has been developed, for example, for four points that are purported to be the orthographic projection of the vertices of a cube.

7.6 Integration

Earlier reports have described our work on using a Markov Random Field model for integrating different visual cues. Recently, Horn has been looking at an alternative methodology for intimate integration of early vision modules. The calculus-of-variations approach to machine vision can be extended to deal with multiple cues by building more complex combinations of "penalty terms". This approach avoids some of the ad hoc methods used to combine outputs from independent modules. The trick is to find a functional to leads to a stable, convergent scheme for solution.

One such project is that of Clay Thompson on integration of binocular stereo and shading under funding from NASA. Stereo and shading have complementary properties: situations where one is weak often are just those where the other is strong. Shading cannot give absolute height or very low spatial frequency components - Stereo is of no use in featureless areas, smears height features, but is very good at giving absolute height. Unfortunately, finding a good combined functional is not easy, we have looked at about a dozen so far.

7.7 Early vision representations

Steve White has been working on a simple nonlinear model of early vision which preserves important primitive edge feature information such as location, orientation, contrast and focus. The goal has been to integrate this simple model in such problem domains as stereo, motion and object recognition. Stereo has been the primary experimental focus in the past year, with considerable success. Current work is exploring the extension of the approach to model matching. A special area of interest is in the possibility of this model being consistent with cortical processing. A significant amount of experimental evidence ranging from simple cell receptive fields and shunting inhibition properties to stereo near/far and tuned representations and other complex cell properties seems to be highly encouraging that something very much like the proposed model of early vision processing may be similar to cortical processing. Details of the approach are described in a separate paper in the proceedings.

8 Using visual routines

While much of our work has focused on specific visual processing modules, we are also interested in the application of these methods to realistic tasks. For example, some of our earlier work on object recognition has been used in an industrial application, inspecting tubing against a set of specifications. Navigation presents a second opportunity for connecting vision sytems with real world tasks.

In this vein, Ian Horswill [1991] has been focusing on connecting perception to action in the context of concrete, routine activities. To do this, he have been examining particular activities in particular contexts, such as traveling down a corridor or interacting with a person. Given specifications of these activities, one can define specific pieces of perceptual information which will suffice to perform them. For example, the direction of the corridor relative to the direction of travel and the proximity of the walls of the corridor are sufficient to navigate the corridor provided that some other process is avoiding obstacles. The extraction of each of these pieces of information can then be phrased as a formal computational problem which can be studied using the standard tools of computational vision.

Focusing on the extraction of these relatively high level pieces of information has advantages. A broad range of systems may be able to extract the same piece of information in very different ways. For example, special properties of the domain can be taken into account to simplify a computational problem. These properties can be analyzed and cataloged for use in similar problems. Another advantage is that the activity for which a piece of information is needed provides well-defined speed and accuracy requirements, and an operational evaluation criterion with which a particular system may be judged.

To date, four working systems have been developed. Three are useful for navigating in office environments - a simple corridor follower, a system which follows a broad class of moving objects, and a simple fail-safe system for preventing collisions. All of these systems use tacit knowledge of their domain - office buildings with carpets having uniform coloration - to simplify the process of figure/ground separation. The last system is a simple stereo proximity detector for use in collision avoidance. The system is quite efficient because it need only concern itself with a specific depth plane, rather than having to create a dense depth map. All of the above systems run in near real time on stock hardware.

Current work is focused on developing an autonomous robot platform which we plan to program to give primitive "tours" of the laboratory using a number of these simple perceptual systems. The expectation is that such a problem will test the range of applicability of this approach to perception and the degree to which machinery and analysis can be reused between the tasks of extracting different pieces of information.

9 Learning

We have continued to develop the theory of learning from examples that we had described in the last Proceedings and we have substantially extended its applications. As we had discussed, our approach to the problem of learning from examples considers it within a mathematical framework as a problem of approximating a multivariate function from sparse data. We have developed a learning/approximation scheme which builds upon classical approximation theory and which is equivalent to a class of multilayer networks with one "hidden" layer that we call HyperBf networks. These networks (a) are based on a rigorous theory with extensive ties to a large body of classical results in applied mathematics; (b) can be easily interpreted in terms of what the components do; and (c) show good performance in several different domains.

Our main research directions are four:

1. extending the theory

2. developing efficient optimisation algorithms for the type of non-convex problems required by the most general versions of HyperBF networks
3. applying the technique to several different domains
4. exploring the potential implications for neurobiology, especially concerning the biological substrate of object recognition

We will review some highlights in the first three of these directions of research after a brief summary of the technique.

9.1 The HyperBF technique

HyperBf networks (Poggio and Girosi, 1990, 1990a, 1990b, 1990c) are a class of feedforward networks with one layer of hidden units that compute functions of the form:

$$f(\mathbf{x}) = \sum_{\alpha=1}^{n} c_\alpha G(\|\mathbf{x} - \mathbf{t}_\alpha)\|_W) + p(\mathbf{x}) \qquad (3)$$

where G is any conditionally definite positive function (Micchelli, 1986), $p(\mathbf{x})$ is a polynomial of low degree, W is a square matrix and $\|\cdot\|_W$ indicates the following *weighted norm*:

$$\|\mathbf{x}\|_W^2 \equiv W\mathbf{x} \cdot W\mathbf{x} \,. \qquad (4)$$

The coefficients c_α, the "centers" $\mathbf{t}_\alpha$ and the matrix $\mathbf{W}$ are found during the learning stage, by minimizing a measure of the error between the network's prediction and each of the examples. After learning the centers of the basis functions are similar to prototypes, since they are points in the multidimensional input space. Updating the centers during learning is therefore equivalent to modifying the corresponding prototypes and corresponds to task-dependent clustering. Finding the optimal weights $\mathbf{W}$ for the norm is equivalent to transforming appropriately, for instance scaling, the input coordinates and corresponds to task-dependent dimensionality reduction.

9.2 Theory

Our main line of investigation was devoted to the problem of selecting a network architecture, because it is one of the choices that strongly influence the final performances. However, once an architecture has been chosen there are other relevant problems that have to be solved. One of these is related to the fact that in many cases the available data may contain outliers, and standard procedures, as least square estimation, have to be modified in this case. Here we show some results on these two topics.

9.2.1 Architecture Selection

Whenever we want to use some kind of network to solve a problem, two fundamental questions arise: a) how many hidden units? b) which activation function should the hidden units compute? We considered the first question under the assumption of an infinite number of examples. The number of units needed to approximate a function within a certain accuracy depends on the choice of the activation function and on some characteristics of the function that has to be approximated, as its dimensionality and degree of smoothness. For many classical spaces of functions and choices of the activation function the dependence of the number of hidden units on the dimension is exponential, leading to the well known phenomenon of "curse of dimensionality". However, if some constraints are imposed on the target functions, better rates of convergence can be obtained. Using a result by Jones (1990) about the rate of convergence of iterative sequences in Hilbert spaces, we proved (Girosi and Anzellotti, 1991) that there exist classes of functions that can be approximated by a network of n radial units with an L_2 error of order $O(\frac{1}{\sqrt{n}})$. The dimension of the space influences the result only through a moltiplicative constant, and the result is constructive, in the sense that it shows an iterative algorithm that can achieve this rate of convergence. Similar results have been obtained by Barron (1991) for multilayer perceptrons, so this raises the question of the choice of the activation function, on which we did some experimental and theoretical work.

Maruyama, Girosi and Poggio (1991a) have compared in numerical experiments several different activation functions, and therefore different techniques for learning from examples, considered as schemes for approximating multivariate functions from sparse data. In particular they considered multilayer perceptrons with one layer of sigmoidal hidden units, flexible Fourier series, multilayer perceptron with exponential activation functions, Radial Basis Functions, and different forms of HyperBF networks. They have characterized their approximation performance (equivalent to generalization power) according to L_2 and L_∞ measures on sparse data from several different continuous functions of two and more variables, using several different training techniques. All the techniques, except the one using exponential activation functions, performed well in the average, and this led to investigate possible relations between multilayer perceptrons and Generalized Radial Basis Functions (GRBF).

The main point of another project of Maruyama, Girosi and Poggio (1991b) was to show that for normalized inputs, multilayer perceptron networks *are* radial function networks (albeit with a non-standard radial function). This provides an interpretation of the weights w as centers t of the radial function network, and therefore as equivalent to *templates*. This insight may be useful for practical applications, including better initialization procedures for MLP. Maruyama et al. also analyse the relation between the radial functions that correspond to the sigmoid for normalised inputs and well-behaved radial basis functions, such as the Gaussian. In particular, they observed that the radial function associated with the sigmoid is an activation function that is good approximation to Gaussian basis functions for a range of values of the bias parameter. The implication is that a MLP network can always simulate a Gaussian GRBF network (with less parameters) but the converse is true only for certain values of the bias parameter. Numerical experiments indicate that the constraint is not always satisfied in practice by MLP networks trained with back-

propagation. *Multiscale* GRBF networks, on the other hand, can approximate MLP networks with a similar number of parameters.

9.2.2 Dealing with outliers

Given n noisy observations g_i of the same quantity f, it is common use to give an estimate of f by minimising the function $\sum_{i=1}^{n}(g_i - f)^2$. From a statistical point of view this corresponds to computing the Maximum Likelyhood estimate, under the assumption of Gaussian noise. However, it is well known that this choice leads to results that are very sensitive to the presence of outliers in the data. For this reason it has been proposed to minimise functions of the form $\sum_{i=1}^{n} V(g_i - f)$, where V is a function that increases less rapidly than the square. Several choices for V have been proposed and successfully used to obtain "robust" estimates. However, a justification and interpretation for their use is still lacking. We have shown (Girosi, 1991; Girosi, Caprile and Poggio, 1991) that, for a class of functions V, that we call "effective potentials", using these robust estimators corresponds to assuming that our measures are affected by a Gaussian noise whose variance is a random variable with given probability distribution. Depending on the probability distribution of the variance of the noise, different shapes for V are obtained. In (Girosi, 1991) a characterization of the class of effective potentials has been given, in terms of positive definite functions in Hilbert spaces.

9.3 Algorithms

Learning the coefficients c_α, the $\mathbf{W}$ matrix and the $\mathbf{t}_\alpha$, that minimize an error functional of the L^2 type on the set of examples is a non-convex minimization problem. Gradient-descent is probably the simplest approach for attempting to find the solution to this problem. We have explored an even simpler optimization technique that can be successfully used to solve this class of problems (Caprile, Girosi and Poggio, 1991). Our algorithm combines aspects typical of many *genetic algorithms* (Goldberg, 1989), with others typical of random descent techniques (Caprile and Girosi, 1990), as the concept of *adaptive noise*. We have tested the algorithm numerically in a variety of cases, and the results have been compared to the ones obtained by using a standard gradient descent with adaptive step technique. In all the cases considered, the best local minima were found by the nondeterministic algorithm, and preliminary experiments suggest that this may hold true also for a class of minimization problems wider than the one we have considered.

9.4 Applications

We are applying the HyperBF technique to several different domains: 3D object recognition; synthesis of algorithms for early visual tasks, such as hyperacuity tasks; computer graphics; time series analysis; adaptive control; indoor vision-driven autonomous navigation. We briefly discuss three of them:

9.4.1 Object Recognition

Edelman and Poggio (1990) applied the HyperBf technique to the problem of 3D object recognition with promising results. They have been able to synthesize a module that can recognize an object from any viewpoint, after it learns its 3D structure from a small set of 2D perspective views, using the HyperBF network scheme. Their results were obtained with simulated wireframe objects and assumed that the problems of feature extraction and matching were already solved. The problems of occlusions and spurious features were ignored. We have now extended successfully the technique to work with gray level images of real paper clips (Brunelli and Poggio, 1991).

It is interesting to mention that psychophysical experiments by Bulthoff and Edleman carried out on wireframe objects and other objects confirm that "immediate" 3D object recognition in humans seems to be based on a process of 2D views interpolation rather than on the use of 3D models.

We have also began to apply HyperBF networks to the problem of recognizing faces, using a small set of images of any given face as examples. This is under the assumptions that a few views for each person are available to train the network (our estimate for a generic 3D object is between 20 and 100 2D views). The theoretical low limit is two views (for the visible aspect) (Basri and Ullman, 1990; Poggio, 1990a). We have therefore begun work aimed to characterize how recognition from just *one* 2D view may be done if views of other ("prototypical") objects of the same class are available (Poggio, 1991). Clearly one single view of a 3D object (if shading is neglected) does not contain sufficient 3D information. If, however, the object belongs to a class of similar objects (prototypes) of which many views are known, it seems possible to make reasonable extrapolations and to guess correctly other views of the specific object from just one 2D view of it. We are certainly able to recognize faces turned way 20-30 degrees from frontal from just one frontal view, presumably because we exploit our extensive knowledge of the typical 3D structure of faces. At this point one can pose the following problem: *from one 2D view of a 3D object generate other views, exploiting knowledge of views of other objects of the same class.* If this can be done, we can then use Poggio and Edelman's technique - and its extensions - by using the views we have generated as a training set. The point is to generate artificial examples of deformations for the specific object of interest by extracting information about allowed deformations from a set of examples of objects of the same class, using standard approximation techniques. Poggio (1991) discusses under which conditions and definitions of class this goal can be achieved.

9.5 Time series analysis

Jim Hutchinson and Tomaso Poggio are engaged in the study of learning architectures, their parallel implementations, and their applications to large, real world problems in time series prediction. The goals of this work are to investigate the potential of parallel implementations to help with problems of parameter estimation, handling of large problems, and use of previously intractable methods; to assess the applicability and usefulness of various learning architectures to the problem of time series prediction; to determine appropriate ways of

achieving domain specific goals in time series modeling, especially obtaining estimates of model fit (i.e. variance of outputs) and methods for iterating predictions; and to determine appropriate ways of handling domain specific problems in time series modeling, such as limited sample size, embedding a priori structure into the learning architecture, and selecting and transforming useful inputs from a collection.

Results to date from this work are fairly preliminary. We have shown that the Radial Basis Function class of learning methods can efficiently be implemented on the Connection Machine for solving large problems. We have investigated various mechanisms for embedding the time series prediction problem in the Radial Basis Function framework, and have preliminary results that such systems outperform corresponding traditional linear models on an interesting class of financial time series.

9.5.1 Early visual tasks

We are beginning to apply the HyperBF technique to explain the fast acquisition of visual abilities in simple tasks from a few examples of the task. Poggio, Fahle and Edelman (1991) were able to show that networks which solve specific visual tasks, such as the evaluation of spatial relations with hyperacuity precision, can be easily synthesized from a small set of examples, using the Hyperbf technique. This may have significant implications for the interpretations of many psychophysical results in terms of neuronal models. It has also potential practical implications in terms of vision architectures that can learn from a set of examples to perform specific visual tasks such as inspection tasks without explicit *ad hoc* programming.

References

[1] Ivan Bachelder. Contour matching using local affine transformations. AI Lab Memo 1326, Mass. Inst. of Tech., 1991.

[2] A.R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. Tech. Rpt. 58, Univ. Ill., Champaign, IL, March 1991.

[3] R. Basri and S. Ullman. The alignment of objects with smooth surfaces. In *Proc. Int. Conf. on Comp. Vis.*, pages 482–488, 1988.

[4] R. Basri and S. Ullman. Recognition by linear combinations of models. AI Lab Memo 1152, Mass. Inst. of Tech., 1990.

[5] T.M. Breuel. An efficient correspondence based algorithm for 2d and 3d model based recognition. AI Lab Memo 1259, Mass. Inst. of Tech., 1990.

[6] T.M. Breuel. Recognition is polynomial time reducible to verification. AI Lab Memo 1268, Mass. Inst. of Tech., 1990.

[7] R. Brunelli and T. Poggio. Face recognition: features vs. templates. Technical report, I.R.S.T., Povo (IT), 1991.

[8] R. Brunelli and T. Poggio. Hyberbf networks for real object recognition. In *Proc. Int. Joint Conf. Art. Int.*, Sydney, Australia, 1991.

[9] R. Brunelli and T. Poggio. HyperBF networks for gender recognition. In *DARPA IU Workshop*. Morgan Kaufmann, San Mateo, CA, January 1991b.

[10] J. Burns, R. Weiss, and E. Riseman. View variation of point set and line segment features. In *DARPA IU Workshop*, pages 650–659, 1990.

[11] B. Caprile and F. Girosi. A nondeterministic minimisation algorithm. A.I. Memo 1254, AI Lab Memo, Cambridge, MA, September 1990.

[12] B. Caprile, F. Girosi, and T. Poggio. A nondeterministic method for multivariate functions minimisation. In *Architetture Parallele e Reti Neurali*, Vietri sul Mare, Salerno, (IT), May 8-10 1991. (in press).

[13] T.A. Cass. Feature matching for object localisation in the presence of uncertainty. In *Proc. Int. Conf. on Comp. Vis.*, pages 360–364, 1990.

[14] D. Clemens and D. Jacobs. Space and time bounds on model indexing. *IEEE Trans. Patt. Anal. Mach. Intell.*, 13, 1991.

[15] D.T. Clemens. Region-based feature interpretation for recognising 3d models in 2d images. AI Lab Memo 1307, Mass. Inst. of Tech., 1991.

[16] J.M. Domingues. Effortless internal camera calibration using images of spheres. Master's thesis, Mass. Inst. of Tech., 1991.

[17] L. Dron. The multi-scale veto model: A two-stage analog network for edge detection and image reconstruction. AI Lab Memo 1320, Mass. Inst. of Tech., 1991.

[18] S. Edelman and T. Poggio. Bringing the grandmother back into the picture: a memory-based view of object recognition. A.I. Memo 1181, AI Lab Memo, 1990.

[19] S. Edelman and T. Poggio. Models of object recognition. *Current Opinion in Neurobiology*, 1:270–273, 1991.

[20] F. Girosi. Models of noise and robust estimates. A.I. Memo 1287, AI Lab Memo, 1991.

[21] F. Girosi and G. Anselotti. Rates of convergence of approximation by translates. A.I. Memo 1288, AI Lab Memo, 1991. (in press).

[22] F. Girosi and T. Poggio. Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Computation*, 1(4):465–469, 1989.

[23] F. Girosi and T. Poggio. Networks and the best approximation property. *Biological Cybernetics*, 63:169–176, 1990.

[24] F. Girosi and T. Poggio. Regularisation, radial basis functions, and recent extensions of networks for learning. In *Proc. 13th IMACS World Congress*, Trinity College, Dublin, Ireland, July 1991.

[25] F. Girosi, T. Poggio, and B. Caprile. Extensions of a theory of networks for approximation and learning: outliers and negative examples. In R. Lippmann, J. Moody, and D. Touretsky, editors, *Advances in Neural Information Processings Systems 3*. Morgan Kaufmann, 1991.

[26] D.E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, Reading, MA, 1989.

[27] W.E.L. Grimson. *Object Recognition by Computer.* MIT Press, 1991.

[28] W.E.L. Grimson and D.P. Huttenlocher. On the Sensitivity of the Hough Transform for Object Recognition. *IEEE Trans. Patt. Anal. Mach. Intell.*, 12(3):255–274, 1990.

[29] I. Horswill. Characterising adaptation by constraint. In *First Europ. Conf. on Artifical Life*, 1991.

[30] I. Horswill and R. Brooks. Situated vision in a dynamic world: Chasing objects. In *Nat. Conf. Art. Intell.*, 1988.

[31] D.P. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proc. Int. Conf. on Comp. Vis.*, pages 102–111, 1987.

[32] D. Jacobs. Optimal matching of planar models in 3d scenes. In *Proc. Comp. Vis. Patt. Recog.*, pages 269–274, 1991.

[33] L.K. Jones. A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. *Annals of Statistic*, 1990. (to appear).

[34] C. Koch and T. Poggio. Multiplying with synapses and neurons. In T. McKenna, J. Davis, and S.F. Zornetser, editors, *Single Neuron Computation*. Academic Press, 1991, in press.

[35] M. Maruyama, F. Girosi, and T. Poggio. Techniques for learning from examples: Numerical comparisons and approximation power. A.I. Memo 1290, AI Lab Memo, 1991a. (in press).

[36] M. Maruyama, F. Girosi, and T. Poggio. A connection between HBF and MLP. A.I. Memo No. 1291, AI Lab Memo, 1991b. (in press).

[37] I.S. McQuirk. Direct methods for estimating the focus of expansion in analog vlsi. Master's thesis, Mass. Inst. of Tech., 1991.

[38] D.B. Mercer. Optimisation of a curvature-gradient dot-product line and edge detector. Master's thesis, Mass. Inst. of Tech., 1991.

[39] C. A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constr. Approx.*, 2:11–22, 1986.

[40] Y. Moses and S. Ullman. Limitations of non model-based recognition schemes. AI Lab Memo 1301, Mass. Inst. of Tech., 1991.

[41] T. Poggio. A theory of how the brain might work. In *Proc. Cold Spring Harbor meeting on Quantitative Biology and the Brain*, 1990.

[42] T. Poggio. 3d object recognition: on a result by Basri and Ullman. Technical report # 9005–03, IRST, Povo, Italy, 1990a.

[43] T. Poggio. 3D object recognition and prototypes: one 2d view may be sufficient. Technical Report 9107–02, I.R.S.T., Povo, Italy, July 1991.

[44] T. Poggio. Economic models and time series: Learning from examples. *The Tactician, Citicorp Technology Office, New York*, 1991.

[45] T. Poggio and S. Edelman. A network that learns to recognize 3D objects. *Nature*, 343:263–266, 1990.

[46] T. Poggio, M. Fahle, and S. Edelman. Synthesis of visual modules from examples: learning hyperacuity. A.I. Memo No. 1271, AI Lab Memo, January 1991.

[47] T. Poggio and F. Girosi. A theory of networks for approximation and learning. A.I. Memo No. 1140, AI Lab Memo, 1989.

[48] T. Poggio and F. Girosi. Continuous stochastic cellular automata that have a stationary distribution and no detailed balance. A.I. Memo 1168, AI Lab Memo, 1990.

[49] T. Poggio and F. Girosi. Hyperbf: A powerful approximation technique for learning. In Patrick H. Winston and Sarah A. Shellard, editors, *Artificial Intelligence at MIT: Expanding Frontiers, Vol. 1*. M.I.T. Press, Cambridge, MA, 1990.

[50] T. Poggio and F. Girosi. A theory of networks for learning. *Science*, 247:978–982, 1990a.

[51] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), September 1990b.

[52] T. Poggio and F. Girosi. Extension of a theory of networks for approximation and learning: dimensionality reduction and clustering. A.I. Memo 1167, AI Lab Memo, 1990c.

[53] T. Poggio and F. Girosi. Learning algorithms and network architectures. In *Proc. Exploring Brain Functions: Models in Neuroscience*, Berlin, Germany, 1991. Dahlem Konferensen.

[54] T. Poggio, A. Verri, and V. Torre. Does cortical area mst know green theorems? I.R.S.T. Technical Report No. 9008-07, Istituto per la Ricerca Scientifica e Tecnologica, Trento, Italy, 1990.

[55] T. Poggio, A. Verri, and V. Torre. Green theorems and qualitative properties of the optical flow. A.i. memo no. 1289, AI Lab Memo, Cambridge, MA, 1991.

[56] T. Poggio and D. Weinshall. The mit vision machine. In R. Chellappa and A. Jain, editors, *Markov Random Fields: Theory and Applications*. Academic Press, 1991 in press.

[57] T. Poggio et al. The MIT Vision Machine. In Patrick H. Winston and Sarah A. Shellard, editors, *Artificial Intelligence at MIT: Expanding Frontiers, Vol. 2*. M.I.T. Press, Cambridge, MA, 1990.

[58] A. Shashua and S. Ullman. Grouping contours by iterated pairing network. In J. Moody R.P. Lippmann and D. S. Touretsky, editors, *Advances in Neural Information Processing Systems 3*. Morgan Kaufmann, 1991.

[59] A. Shashua and S. Ullman. Structural saliency. In *Proc. Int. Conf. on Comp. Vis.*, pages 321–327, 1991.

[60] J.B. Subirana-Vilanova. The skeleton sketch: Finding salient frames of reference. In *DARPA IU Workshop*, pages 399–414. Morgan-Kaufman, 1990.

[61] J.B. Subirana-Vilanova. On contour texture. In *Proc. Comp. Vis. Patt. Recog.*, pages 753–754, 1991.

[62] T.F. Syeda. Finding distinctive color regions in images. In *Proc. SPIE Conf. on Intelligent Robots and Computer Vision IX: Algorithms and Techniques*, 1990.

[63] T.F. Syeda. Data and model-driven selection using color regions. AI Lab Memo 1270, Mass. Inst. of Tech., 1991.

[64] M. Ali Taalebineshaad. Direct recovery of motion and shape in the general case by fixation. In *Proc. Int. Conf. on Comp. Vis.*, 1990.

[65] M. Ali Taalebineshaad. Partial implementation of the fixation method on real images. In *Proc. Comp. Vis. Patt. Recog.*, 1991.

[66] C. Weems, C. Brown, J. Webb, T. Poggio, and J. Kender. Parallel processing in the darpa strategic computing vision program. *IEEE Expert*, 6:23–38, 1991.

[67] W. Wells. A statistical approach to model matching. In *Symposium on Advances in Intelligent Systems*, 1990.

Image Understanding Research at Columbia University

Peter K. Allen John R. Kender Terrance E. Boult Shree K. Nayar*
Center for Research in Intelligent Systems
Department of Computer Science
Columbia University, New York, NY 10027

1 INTRODUCTION

The past year has been a very productive one for the researchers in the Vision/Robotics Laboratory at Columbia University. Since our last report, we have added a new faculty member, Shree K. Nayar, who has continued his work in physics-based vision at Columbia. The coming year heralds a new location and affiliation for the Vision/Robotics laboratory. We will be moving to a new building on the Columbia Campus, the Schapiro Center for Engineering and Physical Sciences Research. We will effectively be doubling our current space, and moving to entirely new and up-to-date Laboratory space. A part of this move, we have formed the new Center for Research in Intelligent Systems (CRIS) which will be focusing on research in perception and cognition, with Image Understanding being a key component of the Center's research agenda. Our fellow faculty members in the Center include vision researchers, roboticists, AI researchers, and Computer Graphics researchers. This is a multi-disciplinary center that will strive to integrate results from each of these fields.

In the sections below, we will outline the work that has occurred over the last year in the following areas of research:

1. Physics-Based Vision
2. Low-Level and Middle-Level Vision
3. Shape Recovery Methods
4. Real-Time Vision
5. Sensor Planning and Modeling
6. Topological Navigation
7. Dexterous Robotic Hands

Technical details can be found in the references and in the papers in this volume from Columbia's researchers.

*This work was supported in part by DARPA contract N00039-84-C-0165.

2 PHYSICS-BASED VISION

2.1 Extending POLARIS

The POLARIS system, which uses polarization information for various vision tasks, has been under development at Columbia for the past 3 years. Last year we extended POLARIS to include edge labeling. This was added to the system previous abilities which were: near-real-time material surface classification, separation of highlight/diffuse components of an scene, and local surface orientation. The physical underpinnings of POLARIS, as well as some of its capabilities can be found in [Wolff and Boult, 1991].

In [Boult and Wolff, 1991] we presented a technique that relies on polarization information to distinguish among 3 types of commonly occurring edges:

occluding or limb edges which are edges produced when the surface turns smoothly away, (i.e. the surface normals become nearly orthogonal to viewing direction as in the sides of a cylinder),

specular edges which are produced from specular reflections of an incident intensity edge, e.g., reflection of light source boundaries or "edges" in secondary light sources, and

albedo/physical edges which are a combination of all other edges including image edges produced from albedo variations, sharp surface discontinuities, and shadows.

The method has been demonstrated on laboratory images, and an example can be seen in figure 1.

2.2 Large-Scale Reflectance Model

The Lambertian reflectance assumption is one of the most widely used assumptions in machine vision. We have discovered that Lambertian surfaces with geometric variations exhibit different reflectance characteristics depending on the resolution of the sensor used to image the surface [Nayar 91c]. When the area viewed by a sensor element is large compared to the dimensions of the surface variations, the reflectance properties are, in general, non-Lambertian. We have developed a reflectance model that describes large scale reflectance from Lambertian surfaces. The surface is modeled as a collection of facets, where the area of each facet is small compared to the surface area viewed by individual sensor elements. The geometrical structure of the surface is then described by a probability distribution function for the facet orientations. Using this surface model, we have derived a reflectance model that accounts for geometrical effects such as masking and shadowing by adjacent facets as well as radiometric effects such as interreflections. We are currently conducting experiments to evaluate the performance of this reflectance model.

2.3 Interreflections and Shape Recovery

Points in the scene, when illuminated, reflect light not only in the direction of the sensor but also between themselves. These *interreflections* can appreciably alter the appearance of the scene. All shape-from-intensity methods are based on the assumption that points in the scene are illuminated only by the sources of light; interreflections are assumed not to exist. Consequently, these methods produce erroneous results when applied to concave surfaces.

We have developed an algorithm that recovers accurate shape information in the presence of interreflections. This solution to the interreflection problem is valid for Lambertian surfaces with possibly varying and unknown color. First, the photometric stereo method is applied to the concave surface to obtain pseudo (erroneous) estimates of shape and color. We have shown [Nayar 92b] that the pseudo shape and color, though erroneous, can be mathematically related to the actual shape and color of the surface. A recovery algorithm has been developed that uses this relation to iteratively recover the actual shape and color from the pseudo estimates. Figure 2a shows a gray-colored concave Lambertian surface of constant reflectance (albedo = 0.75), and Figure 2b shows the erroneous shape extracted using photometric stereo. Figure 2c shows the iterative recovery of the actual shape from the pseudo shape.

2.4 Chromatic Aberration Correction via Image Warping

The problem of chromatic aberration arises because different wavelengths of light are refracted differently by the elements of a lens. Unfortunately, this means that the image of an object is blurred and distorted. In color imaging these distortions cause measurable differences between the images. We have recently show how to use our new image reconstruction filters and previous work on image warping to deal with chromatic aberration correction. The technique has been demonstrated and analyzed on 2 test cases. The technique has been directly compared to the active optics approach from CMU and does quite well, see [Boult and Wolberg, 1991], and our paper in these proceedings [Boult and Wolberg, 1992].

3 LOW AND MID-LEVEL VISION

3.1 Motion Segmentation

Our work in this area addresses the problem of motion segmentation using the Singular Value Decomposition of a feature track matrix. The work builds upon the SVD factorization approach pioneered by Tomasi and Kanade at CMU [Tomasi and Kanade, 1990]. In a recent paper [Boult and Brown, 1991] we have shown that, under general assumptions, the number of numerically nonzero singular values can be used to determine the number of motions. Fur-

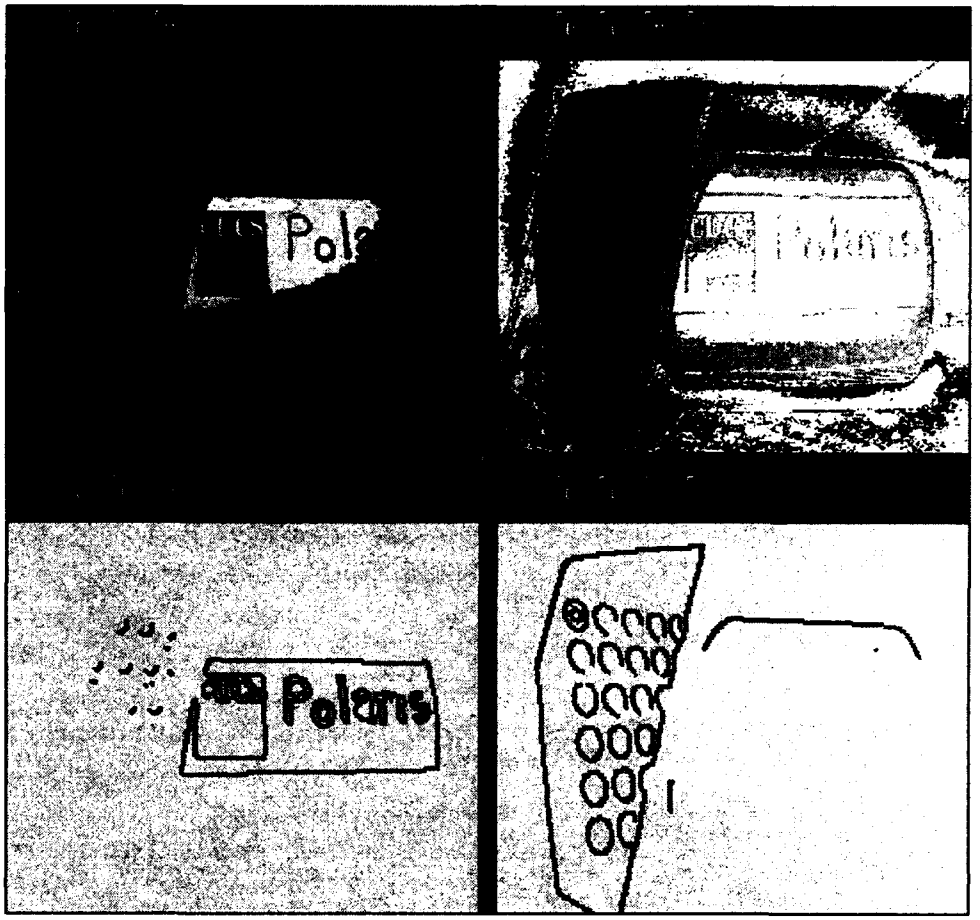


Figure 1: POLARIS edge-labeling for a simple scene. The mug has a paper label on it and the glass plate is specularly reflecting a pattern of circles. Upper Left shows the initial intensity image, upper right shows the percent-polarization (PP) image (white=0%, black $\geq$ 30%). Lower left shows RMS-error edges in black and albedo/physical edges in gray. Lower right shows occluding edges in black and specular edges in gray.

thermore, the motions can be separated using the right singular vectors associated with the nonzero singular values. We also derived a relationship between a good segmentation, the number of nonzero singular values in the input and the sum of the number of nonzero singular values in the segments. The approach has been demonstrated on real and synthetic examples (see the paper in these proceedings [Boult and Brown, 1992].

3.2 Energy-Based Segmentation

Our work in this area is continuing with our most recent results described in [Boult and Lerner, 1990] and [Boult and Lerner, 1991]. Briefly, the technique computes a functional form of the minimal energy surface interpolating a number of data points and also computes a closed form approximation to the bending energy of this surface. It uses heuristics to group points into initial clusters and fits minimal energy surfaces to these seed surfaces. It then grows these surfaces so as to maintain (approximately) minimal energy surfaces, putting points onto those surfaces for which it causes minimal perturbation in the bending-energy. The technique handles overlapping and transparent surfaces; in fact, these are often among the easiest test case examples for it. The technique has now been demonstrated on numerous real and synthetic examples and its being written in the dissertation of M. Lerner[Lerner, 1992].

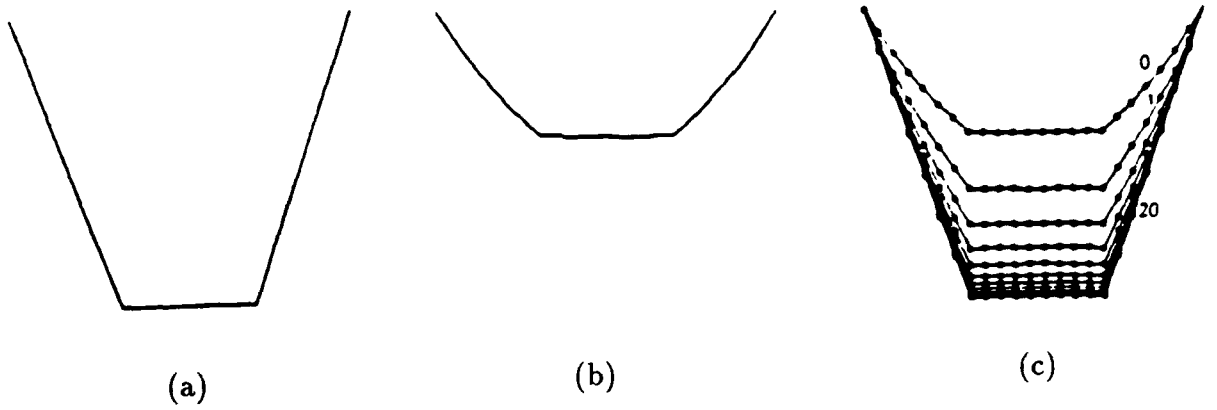


Figure 2: (a) A concave surface; (b) Its pseudo shape extracted using photometric; (c) Recovery of the actual shape from the pseudo shape.

3.3 New models for Image restoration/reconstruction

Using the mathematics of Information-Based Complexity we have recently developed new models for image reconstruction/restoration. These models assume knowledge of the point spread function and the spatial sampling characteristic of the imaging sensor. This spatial sampling results in a blurring of the signal *within a single pixel.* We show how to use a model of this blurring to derive new local reconstruction algorithms with performance equal to or better (depending on your choice of metrics) than previous methods including some global methods like cubic-spline interpolation. The techniques are described in [Wolberg and Boult, 1991].

4 SHAPE RECOVERY METHODS

4.1 Shape from Darkness

The error behavior of the discrete shape from darkness problem [Kender and Smith, 1987] has been explored and formalized. Errorful shadows are shown, under some conditions, to prevent the highly non-linear recovery algorithm from converging. Specifically, non-convergence is seen to be caused by closed loops of shadow constraints that attempt to modify the position of a global reference point. Such loops can be as small as two shadow observations in number, and they have a peculiar grammar defining their character. These loops can be detected in the course of the attempted surface recovery, and the extent (although not the cause) of their error quantified. Algorithms that *attempt to adjust shadow positions minimally,* under various definitions of "minimal", in order to reestablish convergence have been devised and are being tested. The resultant surfaces are therefore near to the expected ideal surface; formalization of expected error in the surface recovery due to these adjustments is being pursued.

4.2 Flexible Extruded Objects

A difficult problem in vision is modeling and recovering the shape of flexible extruded objects such as wires and cables. A Hough-like parameter space technique for modeling flexible extruded objects as piecewise toroidal has been analyzed, and a novel transform has been implemented that derives their three-space curved axes from position and surface normal information. The method is purely local, and succeeds where attempts to model objects as being piecewise cylindrical fail. Although the local computation involves 15 free variables (for three points each: three of position, two of orientation), does not involve the iterative solution of non-linear equations. It has been demonstrated on syn-

thetic and real range data.

Because the torus is an object with seven free parameters, this work also has demonstrated the robustness of the parameter space approach, even for high order objects. Better, it has demonstrated that the structure of the parameter spaces themselves can be chosen to counteract the triangulation error. Errors that occur in trying to find tori in objects that are unusually large or small, or unusually straight or flexed, can be made self-limiting.

This work required the extensive use of a symbolic mathematical analysis system (IBM's proprietary Scratchpad II): the resulting transform is based on a quadratic equation whose coefficients incorporate 12 inner products of three-space vectors. Along the way, it was discovered that, under some fairly general conditions, every torus has a large family of anti-tori. Their hallucinatory appearance in the image must be explicitly ignored.

Most recently, the method has been examined for an extension to cases where the underlying space curve exhibits torsion, since the absence of considerations for torsion in the method appear to be the principal cause of errors in its practical application. If a torus is viewed as a circle swept over the simplest space curve exhibiting curvature, then a "fat helix" (technically, a canal surface) is what results when a circle is swept over the simplest space curve exhibiting torsion. Unfortunately, it currently appears that the mathematical approach used in the existing method depends so heavily on the theorem of Meusnier in differential geometry, that no simple extension with straightforward geometric explanation is possible. Partly the problem is exacerbated by the fact that under general conditions helices are not uniquely defined by a subset of the points they pass through (unlike circles, which are unique given three member points). Less intuitive approaches, based on heuristic restraints on the types of helices that an algorithm should recover, are being pursued instead [Kender and Kjeldsen, 1991, Kender and Kjeldsen, 1992].

4.3 Recovery of SHGCs and Symmetry Analysis

In the previous Image Understanding Workshop, we reported initial results for recovering generalized cylinders; in particular we derived the mathematical constraints obtainable from contours. We also showed how to recover the remaining free parameters from intensity images. This work is described more completely in [Gross, 1991]. Recently, we have been examining the use of heuristics to recover the free parameters. Chief among these heuristics was symmetry of the cross-section because symmetry is pervasive in both man-made objects and nature. Since symmetries project to skew symmetries, finding axes of skew symmetry is an important vision task. Thus we recently developed SYMAN, a SYMmetry ANalyzer, a brief description of which appeared in [Gross and Boult, 1991]. In that paper, we motivated SYMAN's combination of global and local methods. We showed how to derive a global analytic solution for the skew axes when the degree of skew symmetry is known. We also briefly presented a new local tangent-based method which has advantages over previous methods, and gave examples using SYMAN on both real and synthetic images. A more complete description of SYMAN, and derivation of the formulas can be found in [Gross, 1991]. Recent work on SYMAN includes a critical comparison of 4 different techniques for the recovery of skew-symmetry axes [Boult *et al.*, 1991], and the recovery of the orientation of symmetric objects (i.e. complete resolution of skew-axes ambiguity) by the "perceptual grouping" of potential solutions for the underlying multiple orthogonal skew-axes pairs. For example, the pose of the car in figure 3 was obtained using the contours shown, computing their potential skew axes, grouping on these axes and resolving the ambiguity to obtain a global coordinate system, the cars pose.

4.4 Shape from Focus

Shape recovery methods such as stereo, structured light, and shape from shading are effective when applied to smooth diffuse surfaces but produce sparse shape information when applied to rough textured surfaces. Surfaces that

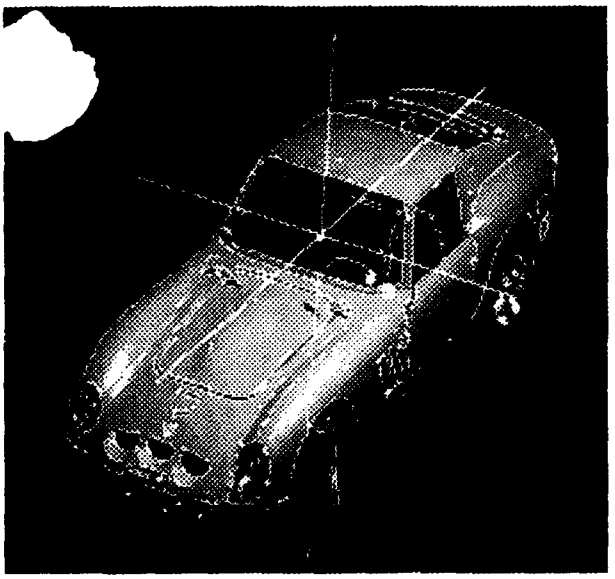


Figure 3: Figure showing the determined pose of a car. The edges shown were used in determining this pose.

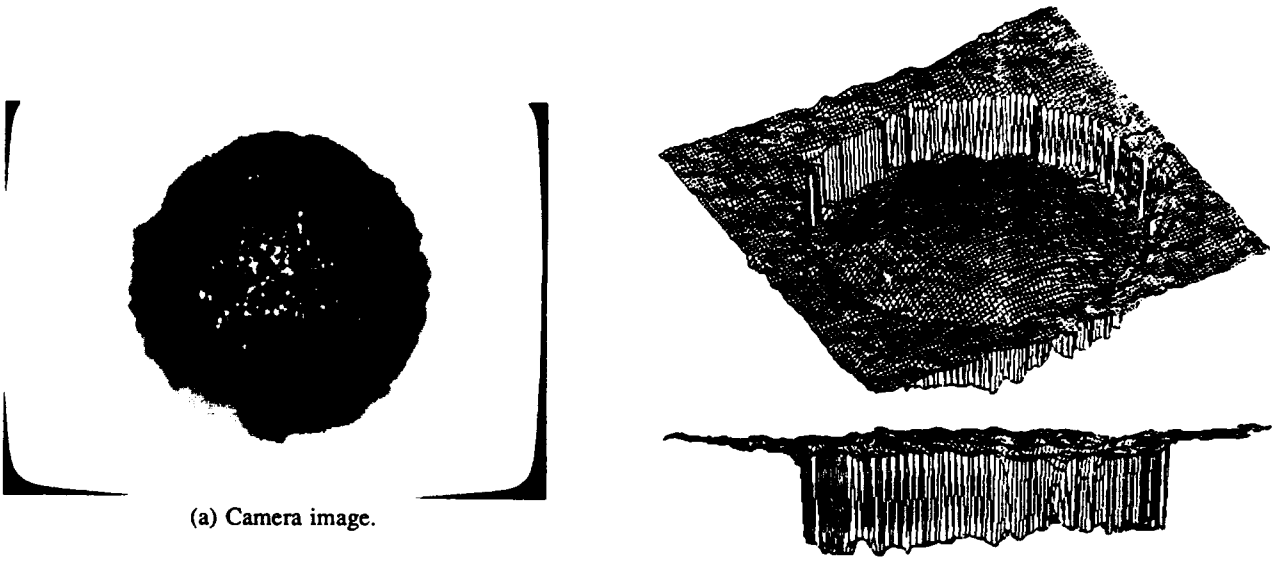
appear smooth on a macroscopic scale are often rough at a microscope scale. Images of rough surfaces are characterized by high frequency intensity variations, and it is difficult to perceive the shapes of these surfaces from their images. We have developed a shape-from-focus method [Nayar 92a] that uses different focus levels to obtain a sequence of object images. The sum-modified-Laplacian (SML) operator is developed to compute local measures of the quality of image focus. The SML operator is applied to the image sequence, and the focus measures obtained at each image point are used to compute local depth estimates. A depth estimation algorithm uses a mathematical model to interpolate the focus measures to obtain accurate depth estimates.

We have recently completed the implementation of a fully automated shape-from-focus system for the recovery of microscopic objects [Nayar 92a]. Figure 4 shows the camera image of a 50 micron via-hole filling on a ceramic substrate and its depth maps recovered by the shape from focus system. We are currently testing the system on biological samples such as micro-organisms and chromosomes.

5 REAL-TIME VISION

The focus of this work is to achieve a high level of interaction between a real-time vision system capable of tracking moving objects in 3-D and a robot arm equipped with a dexterous hand that can be used pick up a moving object. We are interested in exploring the interplay of hand-eye coordination for dynamic grasping tasks such as grasping of parts on a moving conveyor system, assembly of articulated parts or for navigation and grasping from a mobile robotic system. Coordination between an organism's sensing modalities and motor control system is a hallmark of intelligent behavior, and we are pursuing the goal of building an integrated sensing and actuation system that can operate in dynamic as opposed to static environments. The system we have built addresses three distinct problems in robotic hand-eye coordination for grasping moving objects: fast computation of 3-D motion parameters from vision, predictive control of a moving robotic arm to track a moving object, and grasp planning. The system is able to operate at approximately human arm movement rates, and has been demonstrated experimentally by tracking a moving model train, stably grasping it, and picking it up. The algorithms we have developed that relate sensing to actuation are quite general and applicable to a variety of complex robotic tasks that require visual feedback for arm and hand control [Allen *et al.*, 1990b, Allen *et al.*, 1991, Allen *et al.*, 1992].

The vision system uses calibrated but unregistered stereo cameras to track moving objects in 3-D. The images from the cameras are processed by a PIPE image processing engine that performs an optic-flow computation in real-time. The computed velocity fields are morphologically thinned and thresholded on velocity to find regions of object motion. These regions are then triangulated to give a 3-D position vector in less than 100 milliseconds. This 3-D position is sent to the motion tracking algorithm which uses a second order digital filter to smooth the trajectory and a predictive alpha-beta-gamma filter to account for video processing delays. Using this system, a moving object with an ar-



(a) Camera image.

(b) Depth maps.

Figure 4: (a) Camera image of a 50 micron via-hole; (b) Depth map recovered by shape from focus.

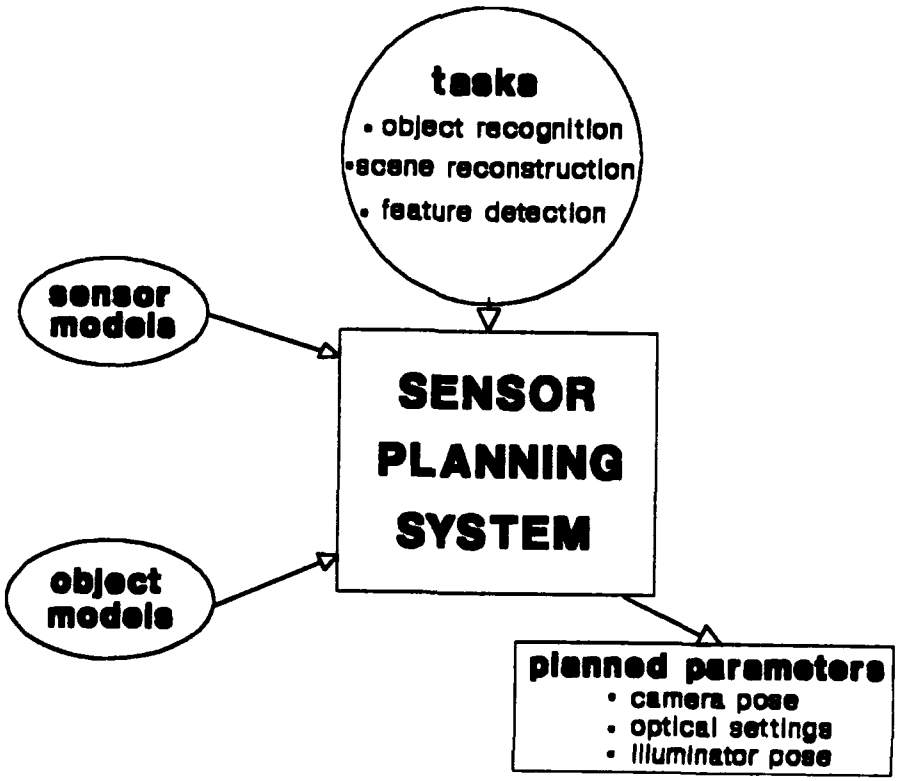


Figure 5: MVP Sensor Planning System.

bitrary trajectory can be tracked in real-time. Once tracking is stable, the system commands the arm to intercept the moving object and the hand is used to grasp the object stably and pick it up. The system has been demonstrated with a variety of trajectories and a video of its performance is available.

6 SENSOR PLANNING AND MODELING

6.1 MVP: Model-Based Sensor Planning for Machine Vision Tasks

In this research we have developed techniques to analytically determine the complete locus of camera poses and optical settings that satisfy visibility, field-of-view, resolution and focus requirements of a machine vision task for given features of interest. This work is part of more extensive research that we are pursuing, as part of our MVP (Machine Vision Planning) system [Tarabanis, 1991a], on the problem of sensor planning for satisfaction of several generic machine vision requirements. We take a synthesis approach to this problem; that is, the admissible domain of sensor locations and settings is determined for each task constraint and then these component results are combined in order to find globally admissible sensor parameter values. This approach improves on the generate-and-test techniques currently employed in which sensor configurations are generated and then tested for satisfaction of the task criteria. In this work, all five degrees of freedom of camera placement are considered and thus the results are applicable to a general three-dimensional viewing configuration. Camera placement experiments have been shown that demonstrate the method in an actual robotic setup. A camera equipped with a programmable zoom lens is mounted on a robotic arm and placed and focused according to the results of the new technique. Camera views are taken to verify that the feature of interest is visible, within the camera field of view and resolvable to the given specification. Results of this research will help automate the vision system design process, assist in programming the vision system itself and lead to intelligent automated robot imaging systems. The object models and a new visibility volumes algorithms have been implemented with ACIS, a commercial solid modeling system. Current efforts are underway to improve the optimization of camera settings using three different techniques: 1) interval analysis, 2) non-linear optimization routines and 3) tree-annealing [Tarabanis *et al.*, 1991a, Tarabanis *et al.*, 1991b, Tarabanis, 1991b, Tarabanis and Tsai, 1991].

6.2 Extending the MVP System

The object and environment models which MVP uses are static CAD models, which means MVP can not plan viewpoints in the face of moving objects in the environment. We are exploring methods of extending MVP to handle motion. The first approach, currently limited to the case of moving obstacles (the target, or features to view, are stationary), is to sweep the model of all moving objects along their trajectories and to plan around the swept volumes, as opposed to the actual objects. This approach has successfully been implemented. A second approach which has not yet been implemented, is to make time an explicit parameter in the CAD models we are using, and to plan a viewpoint that moves in 4 dimensions (monotonic in time), maintaining a robust viewpoint at all times [Abrams and Allen, 1992].

6.3 PROVER

The application of numeric methods to the minimization of error has become an emerging paradigm for object recovery. Typically, a parametric representation describing the object is postulated. Its parameters are then adjusted to minimize some measurement of the distance between the representation and the datapoints (the error-of-fit model). Characteristics of the sensor used to recover the points may be implicit in this formulation or may not be included at all. While sensors may be precise for a specific field of view no sensor is everywhere exact. A laser range finder for example, yields very sharp x- and y-coordinate values; however, its z-coordinate is less trustworthy. It becomes important to capture the strengths and weaknesses of a sensor and incorporate them into the recovery process. We seek to make explicit the contribution of a particular sensor by introduc-

ing a sensor model. This partitioning facilitates the development of an appropriate description of a sensor's characteristics. Also, it helps clarify interactions among different aspects of the recovery process (i.e. error-of-fit model, sensor model, and parametric object representation). The sensor model is reflected in the certainty of sensed quantities (position, color, intensity) associated with a datapoint. We explore whether the introduction of an explicit sensor model yields an improvement in the recovery process. The PROVER (Parametric Recovery Of Volumes: Experimental Recovery) system is a testbed used in the development of sensor models as described in [O'Donnell and Boult, 1991]. As shown in figure 6, this object oriented distributed system takes a number of modules and uses them to recover parametric representations. We can then compare the results of these recoveries to analyze the advantages of different sensor models.

7 TOPOLOGICAL NAVIGATION

The model for topological visual navigation in two-dimensional spaces, already formalized and demonstrated, has been extended by further analyses of the components of qualitative direction-giving.

The efficiency with which an object can be used as part of a series of directions (that is, the extent to which an object is a "landmark") depends heavily on several properties. An important one is the relative object clutter of the individual images taken of the environment; landmarks thus depend upon the size of the viewing aperture.

At least four such aperture-dependent properties have been formalized, and their navigation and description costs are being evaluated in a full navigation system. First, a landmark is considered "obvious" if it uniquely occurs in an image of the navigational world (under a fairly wide set of viewing circumstances). Second, a landmark is "confusable" with another landmark if an arbitrary choice of landmark in an image has no effect on the result of the navigation. Third, a landmark is "new" if it can be distinguished from identical landmarks in prior images; landmarks can thus be path-dependent. Lastly, a landmark may be defined in terms of its spatial configuration with other landmarks, including such virtual landmarks as the boundaries of the viewing window. Each of these properties can be defined without reference to position, orientation, scale, or magic numbers, and each leads to particularly simple sets of navigational direction-giving.

Further investigation of the properties of the directions themselves reveal at least two general strategies for navigation. These two strategies appear to be members of a continuum of strategies that can be mixed and matched at will in order to minimize the various costs of navigation: sensing cost, travel cost, and direction-giving cost. The two strategies that have been formalized are based on the concept of a navigational "parkway," a particularly straightforward spatial arrangement of landmarks. (Common physical streets and highways are special cases of parkways.) Navigation may require the controlled departure from one parkway to another; the careful selection of such "trajectories" establishes a second strategy. (Common ocean crossings are a special case of trajectories.) The definition, analysis, generalization, implementation, and evaluation of these and other high-level strategies for metric-free qualitative navigation is proceeding [Kender *et al.*, 1990].

Related to the above work, mathematical techniques from computational geometry and fuzzy set theory have been used to develop a method for the qualitative description of the topological properties of a set of significant objects or landmarks. The method permits the fuzzy description of a spatial array of objects, independent of metric properties or magic number thresholds.

The method constructs the convex hull of a set of points, and uses membership functions from fuzzy set theory to classify the spatial array into such basic geometric shapes as triangles or squares, or into other useful spatial configurations that have no simple English names, such as "pointed polygon". The method is generic, recursively applicable, extensible to three di-

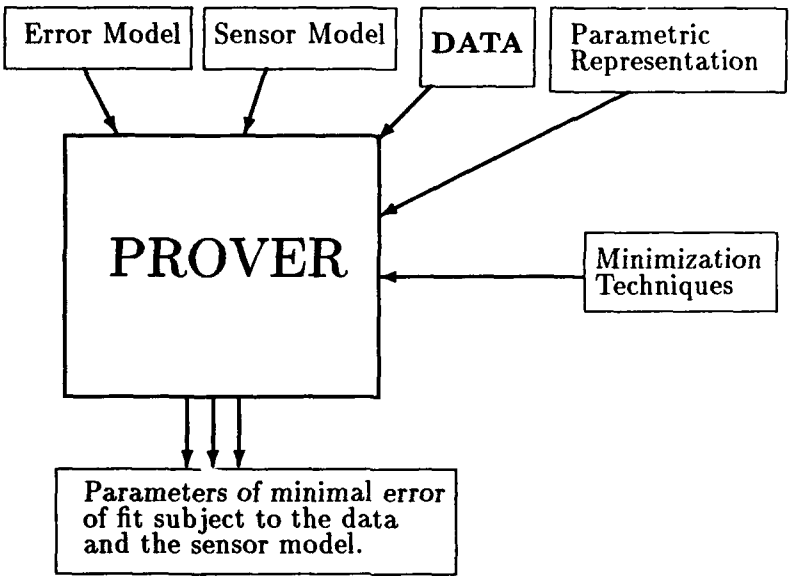


Figure 6: The PROVER system

mensions, and strongly suggests other descriptive categories that are useful even if not recognized by the nouns of a given natural language. The determination of the stability of such descriptions under small perturbations of the spatial configuration is under investigation.

Results from this effort are being integrated with the topological navigation techniques, with which they form a continuum. At one extreme, the world is seen in a series of sensations, and a goal must be attained by a series of directions. At the other extreme, the entire world can seen at a glance, and a goal need only be described ([Abella, 1992]).

8 DEXTEROUS ROBOTIC HANDS

The focus of this work is in exploiting the capabilities of the Utah-MIT dexterous hand. This is a 4-fingered, 16 DOF device with rich sensing (force, position, tactile) and high dexterity. [Allen *et al.*, 1990a, Allen, 1990a]. We include this work in the Image Understanding report because of our desire to integrate both vision and touch in an active sensing paradigm.

8.1 Haptic Object Recognition

Object recognition has focused on vision based methods. Humans also have a very highly developed haptic perception system. By haptic, we mean the interplay of both the cutaneous system (skin, tactile receptors) and the kinaesthetic system (joints, muscle and bone). We have developed robotic analogs of 3 human haptic sensing strategies to recover the 3D shape of objects using our hand/arm system [Allen and Michelman, 1990]. Vision sensing is also being added to allow fully autonomous object recognition. The idea here is that complementary sensing form both vision and touch allows for more robust and stable recognition. The vision consists of a real-time linear feature extractor that is input to a robust line-based stereo system that recovers 3-D axes of surfaces of revolution. These axes are then used by the hand system to explore the object's contour and recover the shape of the object [Allen, 1990b, Singh and Shneier, 1990]. A related work by Kenneth Roberts [Roberts, 1991] uses active touch exploration to recognize a 3-D object taken from a known set of models. This work combines three approaches: (1) using geometric constraints between components to eliminate interpretations, (2) interpretation tree methods for choosing the best active sensing move, (3) exploratory moves made by tracing continually along the surface of the object (and not through free space). This method uses a set of polyhedral object models and exploits a set of geometric constraints tailored for matching compo-

nents acquired from haptic exploration against components in the models. A new constraint using pairs of line segments has been developed and this constraint is also relevant to problems in computer vision. An interesting aspect of this work is that active touch sensing moves are associated with costs, and this allows the determination of sensing strategies and methods for choosing the next sensor move for recognition tasks.

8.2 Tool Usage with a Dexterous Hand

This research investigates the ways in which tools can be used by dexterous robot hands. As robots are called upon to perform finer and more complex tasks, they will begin to use manipulators with a large number of degrees of freedom and sensors. For example, in hazardous environments, such as nuclear reactors, teleoperated robot hands will be used for grasping and manipulation. As the application of robotics extends more toward space, the long communication time delays will necessitate imparting greater autonomy to robots in fine manipulation tasks.

The tasks we have been studying are precision tool tasks, such as using a fine screwdriver, a pair of tweezers, a pencil or an eraser. The significance of these precision tasks is that the tool—the object being manipulated—is controlled strictly by the fingers of the hand, rather than by a wrist or arm. Although much of our work has been in analyzing the requirements and mechanics of manipulation tasks, our ultimate goal is the implementation of the the tool tasks cited above on a Utah/MIT dexterous hand. Therefore, we have also worked on the development of digital position and force controllers for a robot hand, kinematic calibration of the hand, and tactile and force sensing for the robot's fingers [Michelman and Allen, 1991, Jiang *et al.*, 1991]

References

[Abella, 1992] A. Abella. Extracting geometric shapes from a set of points. In *Proceedings of the DARPA Image Understanding Workshop*, Jan. 1992. (These proceedings.).

[Abrams and Allen, 1992] Steven Abrams and Peter K. Allen. Sensor planning in an active robotic work cell. In *Proc. DARPA Image Understanding Workshop*, January 1992.

[Allen and Michelman, 1990] Peter K. Allen and Paul Michelman. Acquisition and interpretation of 3-D sensor data from touch. *IEEE Transactions on Robotics and Automation*, pages 379–404, August 1990.

[Allen *et al.*, 1990a] Peter Allen, Paul Michelman, and Kenneths S. Roberts. A system for programming and controlling a multi-sensor robotic hand. *IEEE Transactions of Systems, Man and Cybernetics*, November/December 1990.

[Allen *et al.*, 1990b] Peter K. Allen, Bil Yoshimi, and Alex Timcenko. Real-time visual servoing. In *Proceedings DARPA Image Understanding Workshop*, Pittsburgh, Sept. 11–13 1990.

[Allen *et al.*, 1991] Peter K. Allen, Billibon Yoshimi, Alex Timcenko, and Paul Michelman. Automated tracking and grasping of a moving object. Technical Report CUCS-034-091, Department of Computer Science, Columbia University, 1991.

[Allen *et al.*, 1992] Peter K. Allen, Billibon Yoshimi, Alex Timcenko, and Paul Michelman. Trajectory filtering and prediction for automated tracking and grasping of a moving object. In *Proceedings DARPA Image Understanding Workshop*, January 1992.

[Allen, 1990a] Peter K. Allen. Active sensing with a dextrous robotic hand. In *NATO Workshop on Traditional and Non-Traditional Robotic Sensors*, volume F 63, Maratea, Italy, 1990. NATO ASI series, Springer-Verlag.

[Allen, 1990b] Peter K. Allen. Mapping haptic exploratory procedures to multiple shape representations. In *1990 IEEE International Conference on Robotics & Automation*, pages 1679–1685, May 13-18 1990.

[Boult and Brown, 1991] T.E. Boult and L.G. Brown. Factorization-based segmentation of motions. In *Proceedings of the IEEE Computer Society Workshop on Motion Understanding*. IEEE, October 1991.

[Boult and Brown, 1992] T.E. Boult and L.G. Brown. Motion segmentation using singular value decomposition. In *DARPA Image Understanding Workshop*, January 1992.

[Boult and Lerner, 1990] T.E. Boult and M.L. Lerner. Energy-based segmentation of sparse range surfaces. In *Proceedings of the SPIE Conference on Sesnor Fusion*. SPIE, November 1990.

[Boult and Lerner, 1991] T.E. Boult and M.A. Lerner. Energy-based segmentation of sparse rande data. In A.L. Méhauté P.J. Laurent and L. L. Schumaker, editors, *Curves and Surfaces*, pages 43–50. Academic Press, NYC, NY, 1991.

[Boult and Wolberg, 1991] T.E. Boult and G. Wolberg. Correcting chromatic aberrations via image warping. Technical report, Department of Computer Science, Columbia University, 1991.

[Boult and Wolberg, 1992] T.E. Boult and G. Wolberg. Correcting chromatic aberrations via image warping. In *DARPA Image Understanding Workshop*, January 1992.

[Boult and Wolff, 1991] T.E. Boult and L.B. Wolff. Physically-based edge labeling. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1991.

[Boult *et al.*, 1991] T.E. Boult, A.D. Gross, and H. Soejhada. Skewed symmetry detection: An evaluation of the state of the art. Technical report, Department of Computer Science, Columbia University, 1991.

[Gross and Boult, 1991] A.D. Gross and T.E. Boult. Syman: a symmetry analyzer. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1991.

[Gross, 1991] A.D. Gross. *Shape Constraints from Parametric and Non-parametric Models*. PhD thesis, Dept. of Computer Science, Columbia Univ., Fall 1991.

[Jiang *et al.*, 1991] J. C. Jiang, R. C. White, and Peter K. Allen. Micro cavity vacuum pressure sensors for robot tactile sensing. In *Proc. of Int. Conf. on Solid-State Sensors and Actuators*, June 1991.

[Kender and Kjeldsen, 1991] J. R. Kender and R. Kjeldsen. On seeing spaghetti: A novel self-adjusting seven parameter hough space for analyzing flexible extruded objects. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Aug. 1991.

[Kender and Kjeldsen, 1992] J. R. Kender and R. Kjeldsen. On seeing spaghetti: A novel self-adjusting seven parameterhough space for analyzing flexible extruded objects. In *Proceedings of the DARPA Image Understanding Workshop*, Jan. 1992. (These proceedings.).

[Kender and Smith, 1987] J. R. Kender and E. M. Smith. Shape from darkness: Deriving surface information from dynamic shadows. In *ICCV*, pages 539–546, June 1987.

[Kender *et al.*, 1990] J. R. Kender, I. P. Park, and D. Yang. Navigation by landmarks. In *Proceedings of the SPIE Symposium On Advances in Intelligent Systems*, November 1990.

[Lerner, 1992] M. Lerner. *Energy-Basxed Segmentation*. PhD thesis, Dept. of Computer Science, Columbia Univ., May 1992. (in progress).

[Michelman and Allen, 1991] Paul Michelman and Peter Allen. Tool usage with a dexterous robot hand. In *ANS 4th Topical Meeting on Robotics and Remote Systems, Albuquerque, New Mexico*, February 1991.

[Nayar 90a] S. K. Nayar, Y. Nakagawa, "Shape from Focus: An Effective Approach for Rough Surfaces," *Proc. of IEEE International Conference on Robotics and Automation*, pp. 218-225, May, 1990.

[Nayar 90b] S. K. Nayar, A. C. Sanderson, L. E. Weiss, D. D. Simon, "Specular Surface Inspection Using Structured Highlight and Gaussian Images," *IEEE Tran. on Robotics and Automation*, Vol. 6, No. 2, pp. 208-218, April, 1990.

[Nayar 90c] S. K. Nayar, K. Ikeuchi, T. Kanade, "Determining Shape

and Reflectance of Hybrid Surfaces by Photometric Sampling," *IEEE Trans. on Robotics and Automation*, Vol. 6, No. 4, pp. 418-431, August, 1990.

[Nayar 90d]
S. K. Nayar, K. Ikeuchi, T. Kanade, "Shape from Interreflections," *Proc. of International Conference on Computer Vision*, Vol. 13, No. 7, pp. 2-11, December, 1990.

[Nayar 90e] S. K. Nayar, *Shape Recovery using Physical Models of Reflection and Interreflection*, Ph.D. Dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University, December, 1990.

[Nayar 91a] S. K. Nayar, K. Ikeuchi, T. Kanade, "Surface Reflection: Physical and Geometrical Perspectives," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pp. 611-634, July, 1991.

[Nayar 91b]
S. K. Nayar, K. Ikeuchi, T. Kanade, "Shape from Interreflections," *International Journal of Computer Vision*, Vol. 6, No. 3, pp. 173-195, August, 1991.

[Nayar 91c] S. K. Nayar, M. Oren, "Non-Lambertian Reflectance from Lambertian Surfaces," Technical Report, Department of Computer Science, Columbia University (in preparation).

[Nayar 92a] S. K. Nayar, "Shape from Focus System for Rough Surfaces," *Proc. Image Understanding Workshop*, January, 1992.

[Nayar 92b] S. K. Nayar, Y. Gong, "Colored Interreflections and Shape Revovery," *Proc. Image Understanding Workshop*, January, 1992.

[O'Donnell and Boult, 1991] T. O'Donnell and T.E. Boult. Introduction of explicit sensor models in parametric object recovery. In *Proceedings of the SPIE Conference on Sesnor Fusion*. SPIE, November 1991.

[Roberts, 1991] Kenneth S. Roberts. *Object recognition with a robot hand.* PhD thesis, Department of Computer Science, Columbia University, New York, May 1991.

[Singh and Shneier, 1990] Ajit Singh and M. O. Shneier. A linear feature extraction scheme for PIPE. *Computer Vision, Graphics, and Image Processing*, July 1990.

[Tarabanis and Tsai, 1991] K. Tarabanis and R. Y. Tsai. Computing viewpoints that satisfy optical constraints. In *Proceedings CVPR '91: The Computer Society Conference on Computer Vision and Pattern Recognition*, June 3–6 1991.

[Tarabanis *et al.*, 1991a] K. Tarabanis, Roger Tsai, and Peter K. Allen. Automated sensor planning for robotic vision tasks. In *IEEE International Conference on Robotics and Automation*, Sacramento, April 9–11 1991. (Awarded Anton Philips Prize for best paper).

[Tarabanis *et al.*, 1991b] K. Tarabanis, R.Y. Tsai, and S. Abrams. Planning viewpoints that simultaneously satisfy several feature detectability constraints for robotic vision. In *5th International Conference on Advanced Robotics (ICAR)*, 1991.

[Tarabanis, 1991a] K. Tarabanis. *Sensor planning and modeling for machine vision tasks.* PhD thesis, Department of Computer Science, Columbia University, New York, New York, November 1991.

[Tarabanis, 1991b] K. Tarabanis. Sensor planning in computer vision: a review. Technical report, Computer Science Department, Columbia University, September 1991.

[Tomasi and Kanade, 1990] C. Tomasi and T. Kanade. Shape and motion without depth. In *Proceedings of the IEEE Computer Society International Conference on Computer Vision*, pages 91–95. IEEE, Dec. 1990.

[Wolberg and Boult, 1991] G. Wolberg and T.E. Boult. Imaging consistent reconstruction/restoration and the integrating resampler. Technical report, Department of Computer Science, Columbia University, 1991.

[Wolff and Boult, 1991] L.B. Wolff and T.E. Boult. Using a polarization reflectance model to constrain object features. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), Special Issue on Physical Modeling in Computer Vision*, 13(6), June 1991.

Maryland Progress in Image Understanding

Yiannis Aloimonos **Rama Chellappa**
Larry S. Davis **Azriel Rosenfeld**
Computer Vision Laboratory, Center for Automation Research,
University of Maryland, College Park, MD 20742-3411

Abstract

Research in the Computer Vision Laboratory at Maryland is focused on problems whose solutions would constitute significant progress in image understanding. This paper reviews our work on these problems during the period August 1990 – September 1991. The areas covered include low-level vision, scene recovery, purposive vision, navigation, object recognition, and parallel algorithms, as well as a few other topics.

1 Low-level vision

Two major goals of low-level vision are the identification of regions and region boundaries in an image that arise from surfaces or surface discontinuities in the scene. These tasks of segmentation and edge detection are not straightforward because of the presence of noise, which may arise either in the scene itself or in the imaging process. Smoothing of the image to reduce noise, estimation to identify image regions that are good fits to simple functions, and differentiation to aid in identifying image discontinuities, are commonly used preprocessing operations.

Our research on low-level vision over the past year has involved the development of new methods of differentiation, smoothing, estimation, edge detection, and segmentation. Approaches investigated include Markov random field and neural network techniques; multiresolution ("pyramid") techniques; robust estimation and "consensus" techniques; Bayesian and maximum a posteriori (MAP) techniques; and optimization techniques. We have applied these approaches to various types of images, including aerial photographs, synthetic-aperture radar images, and range images.

1.1 Differentiation [30]

Reliable derivatives of digital images have always been hard to obtain, especially (but not only) at high orders. We have developed new filters that give more accurate derivatives than the traditional Gaussian ones. We have shown that the traditional filters give incorrect derivatives even for an analytic, noiseless, infinite image, because they smooth the image too much. For a finite interval, the effects of truncating the filter become intolerable for high derivatives. We have derived filters that allow a higher amount of noise suppression with less compromise of accuracy than the Gaussian. The filters are easy to compute at arbitrary size. In addition, a general analytic (non-filter) solution has been derived for the regularization problem on a finite interval.

1.2 Smoothing [3, 13]

A non-iterative, parameter-free image smoothing method has been developed which handles both additive and multiplicative noise. First for every pixel the largest centered neighborhood ($7 \times 7, 5 \times 5$ or 3×3) containing no discontinuity is sought. The selection is made by comparing a local discontinuity measure with its robust global estimate corresponding to homogeneous neighborhoods. If no discontinuity is present, the pixel is assigned the spatial mean computed in the neighborhood. Around discontinuities an adaptive least squares smoothing method is applied in 3×3 neighborhoods. The performance of the multiresolution algorithm has been compared with flat-facet smoothing and adaptive smoothing for additive noise. The smoothing of synthetic aperture radar images has been used to demonstrate the effectiveness of the algorithm for multiplicative noise.

Further work on image smoothing has led to a hierarchical implementation of an edge-preserving smoothing algorithm on a 2×2 pyramid structure. The smoothed pixel values are chosen from the first three levels of the pyramid and the original image. The reduced resolution representations are analyzed in a top-down fashion by comparing the local variances with the corresponding global noise variance estimates. The global estimates are also computed in the pyramid. Close to edges the pixel values are obtained by adaptive least squares. The artifacts of region-based smoothing are eliminated by pixelwise averaging over a set of outputs obtained with the input image shifted within the 8×8 block of the level-three parent.

1.3 Robust estimation and consensus methods [12, 31, 55]

When the images of interest can be modeled as simple functions corrupted by noise, estimation of the parameters of these functions provides a powerful approach to

image smoothing and segmentation. For estimation purposes, robust high breakdown point techniques are gaining increasing popularity in computer vision since they can tolerate up to half the data being severely corrupted. The least median of squares (LMedS) estimator is the best known example of such techniques. We have shown that the attractive properties of the LMedS estimator do not hold when all the data is corrupted by zero mean noise. We have developed a "consensus by decomposition" (CBD) algorithm which preserves the properties of LMedS up to low signal-to-noise ratios while achieving a significant speed-up relative to LMedS. The CBD estimator uses a different paradigm than LMedS. The data is decomposed in both the spatial and parameter domains. A separate distribution is built for every parameter and the distribution is analyzed with a new, enhanced mode detection procedure. The superiority of the CBD estimator has been demonstrated by extensive simulations.

The consensus approach constitutes a paradigm for performing robust model fitting. Model fitting is an essential part of many algorithms that try to interpret data. Such algorithms normally use least squares based estimation for fitting purposes. The least squares procedure is very sensitive to noise and variations in the data. Furthermore, it is frequently the case that a model discontinuity exists in the data and it cannot be reliably detected by conventional methods.

Recently, new robust estimation methods have been introduced that deal with several types of variations and model discontinuities. Unfortunately, as already mentioned, analysis shows that some of these methods are not effective in the presence of noise such as is usually found in image and signal data. The consensus paradigm integrates basic robust estimation methods, a decomposition of the problem and a relative majority analysis to perform robust estimation. The resulting robust estimate is a consensus among independent, invariant estimates of sub-problems.

The consensus paradigm provides effective algorithms for signal and image estimation, edge detection, segmentation, and contour matching. The performance of these algorithms has been analyzed and compared to existing methods.

As an example of the consensus approach, we have developed a new robust algorithm for edge detection. The algorithm detects both roof and step type edges. A pixel is declared as an edge pixel if there is a consensus between different processes that try to determine if the pixel lies on a discontinuity. We use robust estimation methods to estimate local fits to windows in the pixel's neighborhood and accumulate votes from each fit. The use of robust estimators enables us to transform any window possibly containing a discontinuity to a binary window containing a step edge in the location of the discontinuity. We then employ conventional methods to detect this step edge. For simulated edges occurring in synthetic images with varying Gaussian and random noise levels, we have analyzed the probability of detection. The algorithm has also been applied to real intensity and range images and shown to perform well in comparison with standard edge detectors. Further details can be found in a paper in these proceedings.

1.4 Bayesian estimation and MAP methods [27, 50, 54, 58]

When the ensemble of ideal scenes and the noise can be modeled probabilistically, Bayesian estimation techniques can in principle be used to find the ideal scene that most likely gave rise to the observed noisy image; this is the Maximum A Posteriori (MAP) estimate of the scene. A common objection to Bayesian estimation is that the probability density functions (pdf's) involved are usually not exactly known. In fact, however, exact knowledge of the pdf's is not important; it often suffices to know the pdf's approximately. Furthermore, it may even suffice if we have a family of pdf's one of which approximates the actual pdf, provided we specify a "second-stage" pdf on the family such that the approximation of the actual pdf has high probability.

Bayesian estimation of digital signals (or images) is ordinarily concerned with the estimation of ideal signal, given a noisy signal. The computational cost of this process is greatly reduced if the objective is only partial or "qualitative" Bayesian *description*, rather than complete estimation, of the ideal signal. For example, in the case of a piecewise constant signal, instead of estimating the value of the ideal signal, we can require only a piecewise symbolic description of the signal—e.g., is the value high or low, where these descriptors are defined by probability densities on the possible signal values. This task is computationally less costly than that of complete Bayesian estimation of the signal; moreover, the descriptions can be estimated robustly. We have demonstrated this both for digital signals and for a simple class of digital images.

The problem of estimation using partial (e.g., compressed) information about the observations is important in practice. One reason for its importance is that we might be interested in communicating data from the sensor(s) to the place where decisions are made (e.g., remote sensing data). Another reason is that estimation using compressed information might be less costly in terms of computation. We have studied the problem of estimating the parameters of a signal of known form (e.g. polynomial of degree r) using a Bayesian approach to estimation. Conditions can be formulated under which the estimates obtained using partial information are the same as those obtained using full information. This approach has applications to distributed detection (sensor fusion). Partial information about the observations can also be used effectively to obtain partial estimates of the signal.

Bayesian methods can be used not only for conventional signal estimation tasks, but also for identification of data generated by formal models. For example, it can be used to recover, from a finite set of candidate grammars, the most probable grammar (and derivation) that generated the non-noisy version of an observed noisy string, where the noise process is iid and defined by an arbitrary stochastic matrix. We have shown that if the grammars are context-free or stochastic context-free, this problem is solvable in polynomial time.

1.5 Optimization-based methods [35, 42]

Optimality criteria are commonly used for the segmentation of images into homogeneous regions. In this tradition, we have developed several algorithms for segmenting highly speckled high resolution Synthetic Aperture Radar (SAR) complex data into spatially and radiometrically homogeneous regions. Our approach is based on two models, one for the speckled complex amplitudes, and the other for the regions. The first model uses the physics of the SAR imaging and processing system to characterize the statistics of speckle while the second model uses a Markov random field to describe the statistics of the regions. Based on the combination of these two models using Bayes' theorem, two possible optimality criteria are used for the segmentation of the complex data into regions. The resulting algorithms are implemented on a parallel optimization network. Results using both simulated and actual SAR complex data have been used to compare the algorithms and evaluate their performance.

Current object recognition systems can only "recognize" a limited class of objects. Objects having variable numbers of parts and only loosely constrained shapes cannot be modeled and recognized by these systems. We developed a data structure called the *VAPOR* (Variable APpearance Object Representation) model to represent objects with these kinds of variable appearances and have developed a search procedure called *MOSS* (MOdel Space Search) to find instances of these models in image data. The VAPOR model is an idealization of the object; all instances of the model in the image are variations from the ideal appearance. The variations are evaluated by the *description length* of the model, measured in information-theoretic bits. MOSS selects the best model for the given image data by choosing the minimal length description. We have demonstrated how this approach performs in a simple domain of circles and polygons and in the complex domain of finding cloverleaf intersections in aerial images of roads.

1.6 Cluster analysis [9, 49]

Both consensus and MAP methods can be applied to the detection of clusters in sparse data. As an illustration of the consensus approach, we have investigated the use of pyramid algorithms for compact region detection and delineation to detect compact dot clusters on a sparsely dotted background. When the delineation process is applied to a detected cluster, it yields "ragged" results which are sensitive to the position of the cluster in the image. But if the process is applied to a set of shifted versions of the cluster, the results can be combined into an acceptable delineation. This provides a further illustration of the general "consensus" approach to combining multiple techniques, or multiple versions of the same technique, to obtain more reliable results.

The problem of dot clustering can also be studied from a model-based viewpoint. In this approach, a set of scatter processes (in brief: scatters) is chosen, each of which associates a probability with each location in a discrete space; in other words, a scatter is a probability mass function (pmf) on the space. Some number of dots is then distributed in accordance with each scatter. A scatter and an associated numerosity define a subpopulation of dots. This model is extremely general; the scatter pmf's are arbitrary. Given a set of dots generated by such a model, we can apply Maximum A Posteriori (MAP) methods to recover the most likely set of scatters and numerosities that could have given rise to the dots. This identification problem is different from the partitioning problem, which asks for the most likely partition of the dot population into subpopulations. MAP methods are especially useful in cluster analysis when the scatters are non-Gaussian. The general identification problem is intractable, but it has a polynomial time solution if the number of clusters is bounded, and a similar result holds for the partitioning problem. The details of this work can be found in a paper in these proceedings.

1.7 Texture analysis [8, 34]

During the past 25 years considerable effort has been expended attempting to formulate a theory of texture segregation and description. Attempts have been made to characterize the information that yields texture segregation in terms of a small set of properties or primitives. These attempts have provided important information concerning the visual processing involved but have not succeeded in establishing a psychophysical theory of texture segregation. Specifying the features yielding texture segregation has proved to be difficult. An alternative approach is to characterize texture segregation in terms of processing mechanisms. Two mechanisms that explain much of the experimental data are spatial-frequency channels and preattentive grouping processes. We have conducted experiments showing that texture segregation can be explained in terms of the differential stimulation of spatial-frequency channels operating on intensity values. However, texture segregation also occurs as a result of preattentive grouping processes. The grouping of discrete elements into a line-like pattern through edge alignment segregates the pattern from surrounding elements. The grouping of intermixed light and dark elements through lightness similarity segregates the pattern into subpopulations. Perceived population segregation is approximately a single-valued function of the differences in the perceived lightnesses of the elements whereas the relevant variable for spatial-frequency channels is stimulus contrast. Recent experiments indicate that texture segregation may be affected by the stimulus representation. We have performed experiments showing that a change in the orientation of a stimulus that keeps the slopes of the component features constant yields stronger texture segregation in a 3D representation (i.e., the figures were seen as three-dimensional) than in a 2D representation (i.e., the figures that were seen as two-dimensional). We believe that the greater texture segregation with a 3D representation is a consequence of grouping processes. A 3D representation makes evident the orientation of object surfaces enabling the grouping of objects by their similarity of surface orientation, e.g., the direction of their surface normals. We conjecture that these grouping processes, because they are based on the 3D interpretation of projected shapes,

require attention.

We have also investigated the problem of texture discrimination in range images. The range data is transformed to a common coordinate system, and then resampled to generate a regular grid of points in order to eliminate the effect of different sampling rates. Statistical textural features based on co-occurrence matrices are computed on the resampled data. These features can be used to discriminate between classes of natural surfaces. Our experiments used surfaces composed of pebbles of different sizes lying on a plane. Many of the standard statistical textural features have been found to be useful in discriminating between such surfaces. The experiments also confirmed the importance of resampling the data before computing the textural features.

1.8 Markov random fields and neural networks [28]

There are significant relationships between neural network and Markov Random Field approaches to low-level vision problems. We have investigated connections between these areas as they apply to image estimation and segmentation, as well as to texture classification and segmentation of textured images.

2 Recovery

We are continuing our work on recovery, i.e. on the theoretical study of the range of possible capabilities that vision systems could have. Our work has concentrated on the analysis of visual motion and on shape reconstruction from cues such as stereo, shading, etc.

2.1 Analysis of visual motion [4, 19, 38, 57]

It has been argued that identifiable features such as points and lines are the only thing needed for computing motion and scene structure because they carry reliable information, there exist mathematical and computational tools to treat them, and the extensive experience gained in photogrammetry can be tapped. On the other hand, the number of image pixels covered by these features is only a tiny fraction of the total number of pixels in the image. If one uses only the point and line features, the vast majority of the image remains unused. Besides this underutilization, two other problems arise. First, there is no consensus among researchers in computer vision as to what a feature is, either in a rigorous mathematical sense or even in an intuitive practical sense, judging from what different detectors detect as features. There is no general algorithm to detect features and match them. The second problem is that there exists no algorithm that works with both points and lines at the same time and guarantees a unique solution, although there are algorithms for each of them separately. A special case of the theory we have developed is an algorithm that can treat both of them at the same time.

We are also continuing our efforts on examining the stability of the structure from motion problem when we employ correspondence as input, using a statistical approach. In particular, we have studied the inherent ambiguities in recovering 3-D motion information from a single optical flow field. These ambiguities are quantified using the Cramer-Rao lower bound (CRLB), which is a lower bound for the error variances of motion parameter estimates. This performance bound is independent of the motion estimation algorithm, and can always be computed for any arbitrary 3-D motion of a rigid surface by inverting a 5×5 matrix. As a special case, the performance bound for the motion of 3-D rigid planar surfaces has been studied in detail. The dependence of the bound on several factors, such as the underlying motion, surface position, surface orientation, field of view, and density of available pixels, can be derived as closed form expressions. A subset of these results supports Adiv's recent analysis of the inherent ambiguities of motion parameters. For the general motion of an arbitrary surface, it turns out that not every pixel gives information regarding 3-D motion estimation. We have shown that the aperture problem in computing the optical flow restricts the nontrivial information about the 3-D motion to a sparse set of pixels at which both components of the flow velocity are observable. Using computer simulations, we have studied the dependence of the inherent ambiguities on the underlying motion, the field of view, and the number of feature points for motion in front of a nonplanar environment. Also, effects of two smoothing schemes on estimation accuracy have been analyzed. We have shown that introducing a smoothness constraint by fitting local patches to 3-D depths gives lower CRLBs. Not surprisingly, this reduction of CRLBs is very small. Further, fitting local patches also relaxes the aperture problem since the motion information is not restricted to the points at which both optical flow components are observable. In contrast, imposing smoothness on the optical flow by regularization does not lower the CRLBs. All in all, our results indicate that robust computation of 3-D motion from noisy optical flow is still far from reality.

We have also developed an algorithm for estimating image motion with motion segmentation in mind. However, computing optical flow remains an ill-posed problem and estimating it very accurately in a general and noisy situation could be impossible. In any case, we need to estimate a visual quantity with an accuracy that will allow us to perform the task at hand. Various tasks may require different accuracies. We have developed an algorithm for estimating optical flow with motion segmentation in mind, i.e. the algorithm is partly guided by a motion segmentation process. Whether its accuracy is enough for tasks like shape from flow or 3-D motion detection is a research issue we are currently investigating.

Image motion is estimated by matching feature "interest" points in different frames of video image sequences. The matching is based on local similarity of the displacement vectors. Clustering in the displacement vector space is used to determine the set of plausible match vectors. Subsequently, a similarity based algorithm performs the actual matching. The feature points are computed using a multiple filter image decomposition operator. The algorithm has been tested on synthetic as well as real video images. The novelty of the approach

consists of the fact that it handles multiple motions and performs motion segmentation.

Over the last five years we have been developing feature based motion estimation methods using a long sequence of image frames. In our previous work we used quaternions to represent the rotational motion. The occlusion of feature points was not addressed. The nonlinear plant model used in the recursive equation also required a time-consuming numerical integration step to update the covariances of the estimated motion and structure parameters. Recently, we have developed a simplified kinematic model based approach with an eye towards developing robust but faster algorithms for 3-D motion estimation. This approach is based on representing the constant translational velocity and constant angular velocity motion using nine rectilinear motion parameters, which are the 3-D vectors of initial position, linear and angular velocities. The structure of the moving object is represented by the coordinates of feature points in a 3-D coordinate system fixed on the object. The measurements are noisy perturbations of 2-D image locations of feature points. A nonlinear least squares method has been developed to compute both estimates of motion and structure parameters using the first few frames. Nonlinear Kalman filters are then used for the incrementational updating of motion and structure parameters. Since the plant models are linear in the new formulation, closed form solutions are possible for feature tracking, thus avoiding the time-consuming numerical integration steps required in the implementation of nonlinear filters. Problems due to occlusion and temporally nonuniform images have also been addressed.

2.2 Transparency and texture [2, 32]

We have continued our work on transparency, motivated by the fact that the extraction of image velocity in the presence of transparency is not accounted for by current motion theories and by their limited ability to discriminate of 3-D texture patterns.

Perceptual transparency, which occurs whenever (i) two or more patterns are *perceived* as lying at different depth levels, and (ii) one pattern is seen *through* other patterns, can be generated by different visual attributes, such as depth from stereo luminance, motion or texture. According to this definition, perceptual transparency can still occur even if the patterns are not physically transparent, or if they partially occlude. We have analyzed how geometrical information can affect the perception of motion transparency and coherence. This information is given by contours or features such as line endpoints or corners. Contours, edges or intensity gradients can correspond to the projection into the image plane of occluding object boundaries, regions of surface curvature inflection, or surface markings, and they constitute an important source of visual information. We have proposed a model for the perception of motion transparency and coherence which is based on a three-stage process for the extraction of image velocity. Our model is different from Hildreth's model for the extraction of image velocity which enforces a smoothness constraint on the image velocity along contours. The three-stage process which describes our model corresponds to a generalization of the two-stage process proposed by Adelson and Movshon; the third stage is given by the accumulation of votes for each velocity vector in the velocity space, and this generates a velocity histogram. The number of prominent peaks in this velocity histogram is related to specific types of motion perception; for two superimposed patterns, we perceive motion transparency, coherence, or mixed motion perception, depending on whether the velocity histogram is bimodal, unimodal or trimodal. The conditions for perceptual motion transparency or coherence, which are given by the ratios between the spread or heights of the velocity histogram peaks, involve the relation between geometrical information, like contour curvature, and (system) noise.

We have also proposed a method for the discrimination of 3-D texture patterns. The visual discrimination of objects made up of small densely distributed opaque elements, e.g. trees and bushes in a forest, is a common task in natural environments. As we walk through a forest we may need to determine the relative positions of the trees and bushes for purposes of navigation or manipulation. Trees and bushes are a subset of what we call 3-D texture, that is, objects defined by the 3-D distribution of texels. These texels can be solid or planar, opaque or transparent.

Current models of texture describe conditions for the discriminability of texture boundaries and grouping, and also treat the reconstruction of surface shape from texture. Nearly all of these models describe texture in terms of the 2-D distribution of texels on smooth, opaque surfaces. 3-D texture poses a challenge to current low-level vision models because it makes it necessary to cope with texel occlusion and transparency. The reconstruction of the shape of a 3-D texture, e.g. a tree or a bush, as well as the computation of the relative distances and depths of 3-D textures, requires the use of prior knowledge. Conventional assumptions, such as the smoothness constraint in regularization theory, or the assumption of local planarity in shape-from-texture models, do not apply to this problem. Trees and bushes are described by three-dimensional (volumetric) distributions of leaves and branches and not by smooth surfaces. Although leaves can be assumed to be approximately flat, and thus to have a given tilt and slant, they are not regularly oriented in space. Tilt and slant are not appropriate variables for describing the shape of a tree or a bush. Motion cues, generated by a moving observer, are important for the discrimination of trees and bushes in a forest. Ordinarily, as described by structure-from-motion models, there exists a one-to-one relation between the 3-D velocity of a (rigid) object in space and its image velocity. On the other hand, when we observe a tree or bush in relative motion most of its leaves are partially or totally occluded. This makes the computation of the velocities of individual leaves difficult, if not impossible.

We have implemented a method for the discrimination of tree-like objects by analyzing the relative motions of piecewise opaque patterns in the fronto-parallel plane. These patterns represent the projections onto the image plane of tree-like objects. For two piecewise opaque

patterns moving across one another the (global) velocity histogram is bi-modal. We achieve "discrimination" by counting the number of prominent peaks in the velocity histogram; this should correspond to the number of tree-like objects in space, provided the objects are not too close to each other.

We assume that the observer has only translational (horizontal or vertical) motion relative to a static scene. By using perspective projection it is easy to verify that the image velocity of a point in the image is equal to the observer velocity. Therefore the difference between the image velocities of leaves and branches belonging to different patterns (trees) is larger than variability of the velocities of leaves and branches belonging to the same pattern. Consequently, the velocity histogram for two superimposed patterns will have prominent peaks; the spread of these two peaks will be generated, in part, by the difference in depth between the frontmost and rearmost leaves and branches within each tree. The ratio of the positions of the two peaks is equal to the ratio of the (average) depths of the two trees.

2.3 Shape from x; stereo [25, 33, 36, 52]

We have developed a new approach to the problem of shape from shading. Assuming uniform albedo and a Lambertian surface for the imaging model, we first estimate the illuminant direction and surface albedo. With the estimated reflectance map parameters, we then compute the surface shape using a new procedure, which implements the smoothness constraint by requiring the gradients of reconstructed intensity to be close to the gradients of the input image. The new algorithm is data driven, stable, updates the surface slope and height maps simultaneously, and significantly reduces the residual errors in irradiance and integrability terms.

In addition, we have designed a feature-based stereo matching system. We use a hierarchical grouping process that groups line segments into more complex structures that are easier to match. The hierarchy consists of lines, vertices, edges and surfaces. Matching starts at the highest level of the hierarchy (surfaces) and proceeds to the lowest (lines). Higher level features are easier to match, because they are fewer in number and more distinct in form. These matches then constraint the matches at lower levels. Perceptual and structural relations are used to group matches into islands of certainty. A Truth Maintenance System (TMS) is used to enforce grouping constraints and eliminate inconsistent match groupings. The TMS is also used for reasoning in the presence of uncertainty and to carry out belief revisions necessitated by additions, deletions and confirmations of hypotheses.

3 Purposive vision [20, 21, 40, 43]

Our efforts on building the Medusa system are continuing on a theoretical basis. We have developed a set of navigational techniques in which the input is the normal optical flow (the spatiotemporal derivatives of the image intensity function) and where full or accurate reconstruction of the visible world is not needed. Inferences about the three-dimensional environment are based on simple image measurements and qualitative strategies employed by an active observer. In particular, we have developed solutions to the problems of egomotion estimation, obstacle avoidance, relative depth computation, estimation of 3-D motion, tracking and detection of independent motion by a moving observer. These solutions are described in a separate paper in these Proceedings.

4 Navigation

Our research on navigation has emphasized two areas: "stealth" path planning on terrain and navigation in an uncertain, dynamic environment.

4.1 Stealth terrain path planning [18, 41, 56]

We are developing data parallel algorithms for planning paths for groups of ground vehicles over natural terrain subject to constraints on navigability, visibility (from adversaries moving through the terrain) and motion protocols (e.g., maintaining line of sight communication between pairs of vehicles). The planning algorithms have been implemented on a CM2. The application of this research to the development of a "bounding overwatch" path planner is described in a paper in these proceedings. Related research includes the development of a data parallel algorithm for constructing a constrained Delaunay triangulation of a digital terrain model. This algorithm has also been implemented on the CM2 and can produce a Delaunay triangulation of a 512×512 digital terrain map in under 70 seconds.

4.2 Navigation with uncertainty [44, 47, 48]

We have developed a probabilistic method for noisy sensor based robotic navigation in dynamic environments. The method generates an optimal trajectory by considering as optimality criteria the probability of not colliding with the obstacles and the probability of accessing an operational position with respect to a moving target object. Estimates of the obstacles' kinematic parameters and measures of confidence in these estimates are used to produce the probability of collision associated with any robot displacement. The probability of collision is derived in two steps: a stochastic model is defined in the kinematic state space of the obstacles, and collision events are given simple geometric characterizations in this state space.

We have also considered the problem of efficient path planning for a point robot in a partially known dynamic environment, where the static, known part of the environment consists of point shelters distributed in planar terrain, and the dynamic, unknown part is abstracted in the form of *alarms* that cause the robot to leave its current (pre-planned) path and divert to the nearest shelter. We have carried out a probabilistic analysis of the expected times for the dynamic paths generated when the alarms follow a Poisson distribution with parameter λ. A case study with three shelters has been performed to illustrate the dependence of the expected travel times on λ for two alternate static paths. Two different strategies have been developed for the general case of n shelters and have been shown to be superior for different ranges

of values of the alarm rate λ (very low and very high values, respectively).

5 Recognition

5.1 Invariants [14, 23]

Invariants of shapes are of major importance in object recognition because they are independent of the point of view from which the shape is seen. However, they require a relatively high number of parameters that need to be extracted from the shape, which raises concerns about the reliability of such descriptors. We have addressed these accuracy and reliability issues and have shown that invariants can be made robust to noise. We have found new differential invariants requiring fewer derivatives; we use smoothing techniques which give much better results than the Gaussian. Our experiments show that the derivatives involved do not pose a serious problem, provided we make the right choice of smoothing method and parameters. Further details can be found in a paper in these proceedings.

For two-dimensional shapes, it turns out to be possible to define approximate invariants that are insensitive to slants of 70° or more. This makes it possible to recognize a two-dimensional shape from a single perspective projection image taken from an unknown (three-dimensional) viewpoint. Our method is based on quadratic approximations to the effect of the slant of an inverse perspective transformation on angles and lengths. These approximations allow us to define contour-based properties that are nearly invariant under perspective transformation. The method can be used to recognize partially occluded shapes, as well as shapes that are not exactly related by perspective transformations. When a shape is recognized, the method also provides estimates of its tilt and slant.

5.2 Pose estimation [24]

Our research on object recognition in range images has focused on a variant of a generalized Hough transform. To obtain data points in the rotation parameter space, pairs of surface normals are chosen at random from the range image. Each pair of surface normals is matched to all pairs of surface normals from the models, provided they make similar angles. Each match defines a rotation axis and a rotation angle between the model and the scene object. To obtain data points in the translation parameter space, triples of range pixels are chosen at random. If the corresponding patches are found to be non-coplanar, these triples of planar patches are matched to triples of faces for the models, provided their surface normals make compatible angles. Each match defines a new point in translation space. A Least Median of Squares (LMS) method is used for clustering in the parameter spaces. The LMS method can accurately estimate the centers of clusters of data points in the presence of outliers on condition that the number of clusters is known, but in this problem the number of clusters is unknown. Therefore, in each parameter space, spheres are considered with their centers at the nodes of a 3-D grid. The LMS method is applied to the data points within the spheres to find cluster centers. The spheres are then recentered at these centers, and the process is repeated until the spheres do not move. The spheres that contain the largest numbers of data points are considered to be centered at the most significant clusters. The final centers are kept as hypotheses for the pose parameters. The hypotheses are verified by a similarity measure.

5.3 Recognition by parts [51]

We have developed an approach to the recovery and recognition of 3-D objects from a single 2-D image. The approach is motivated by the need for more powerful indexing primitives, and shifts the burden of recognition from the model-based verification of simple image features to the bottom-up recovery of complex volumetric primitives. Given a recognition domain consisting of a database of objects, we first select a set of object-centered 3-D volumetric modeling primitives that can be used to construct the objects. Next, using a CAD system, we generate the set of aspects of the primitives. Unlike typical aspect-based recognition systems that use aspects to model entire objects, we use aspects to model the finite set of *parts* from which the objects are constructed. Consequently, the number of aspects is fixed and *independent* of the size of the object database. To accommodate the matching of partial aspects due to primitive occlusion, we introduce a hierarchical aspect representation based on the projected surfaces of the primitives; a set of conditional probabilities captures the ambiguity of mappings between the levels of the hierarchy.

From a region segmentation of the input image, we formulate primitive recovery as the problem of grouping the regions into aspects. No domain dependent heuristics are used; we exploit only the probabilities inherent in the aspect hierarchy. Once the aspects are recovered, we use the aspect hierarchy to infer a set of volumetric primitives and their connectivity relations. Subgraphs of the resulting graph, in which nodes represent 3-D primitives and arcs represent primitive connections, are used as indices to the object database. The verification of object hypotheses consists of a topological verification of the recovered graph, rather than a geometrical verification of image features. A system has been built to demonstrate the approach, and it has been successfully applied to both synthetic and real images.

6 Parallel algorithms for vision and planning [7, 22, 26, 39]

Traditionally, vision and visual navigation have been driving forces behind the development of high performance parallel processing systems. Our research in parallel algorithms for vision and planning focuses on *data parallel algorithms* with two complementary emphases: First, we are concerned with developing algorithms having optimal or near optimal computational and communication complexity, with additional emphasis on the *scalability* of algorithms with respect to architectural factors such as processor set size, memory size, etc. Second, we want our algorithms to be *practically efficient*, which means, on the one hand, that they can be easily imple-

mented using standard data parallel programming constructs (e.g., prefix operations, basic data reorderings, etc.) and, on the other hand, that the algorithms run fast when implemented on scalable parallel processors. In the following subsections, we describe our research on massively parallel algorithms for focus of attention vision and contour data processing.

6.1 Replicated image processing

We are interested in developing data parallel algorithms that will allow us to efficiently process small images on massively parallel computers. Such small images might correspond to tracking windows in a time-varying image sequence, prediction windows in a hypothesize-and-test object recognition system, or high levels in a multiresolution image data structure. We have developed a general methodology for designing such data parallel algorithms, and have illustrated its application to a variety of problems in low level vision including convolution, rank-order filtering, and morphological operations. The methodology is based on *replicating* the image to be processed, and then decomposing the operation to be performed into data parallel components. We have carried out an analysis of replicated image analysis algorithms on a variety of common interconnection networks (2 and 3-D meshes, hypercubes) and identified the conditions (specified in terms of basic architectural parameters) in which replicated algorithms will lead to speedups over conventional data parallel solutions. We have also developed a replicated image analysis algorithm for rank order filtering and have tested it experimentally on both the Connection Machine and Maspar.

6.2 Contour data processing

The analysis of contours plays a fundamental role in many computer vision algorithms (e.g., object recognition, stereo, motion analysis). We have studied problems associated with efficiently processing image contours using data parallel algorithms on massively parallel machines. Traditionally, parallel algorithms for contour processing have operated on the original two dimensional image plane representation. The principal advantage of this is that the spatial relationships between contours are implicitly encoded in the representation. However, there are also significant disadvantages to processing contours embedded in the 2-D image. One disadvantage is that standard data parallel algorithms will spend machine cycles analyzing all of the image pixels that do not lie on contours. Since the number of physical processors in even large parallel machines is far smaller than the number of pixels in even a television frame, this leads to very inefficient algorithms. A second disadvantage is that it is generally not possible to develop asymptotically efficient algorithms for contour processing in the array representation. Specifically, one cannot employ parallel prefix operations, which would result in algorithms having computational complexity that is logarithmic in the length of image contours; instead, most algorithms will be linear in the length of the contours.

We have developed a $\log N$ EREW algorithm (where N is the length of the longest image contour) for transforming image contours from their image plane embedding to a packed, one dimensional data structure and have illustrated the advantages of this representation by presenting $\log N$ algorithms for piecewise approximation, point-in-polygon, and basic local contour feature extraction algorithms (e.g., curvature estimation, corner detection). These algorithms have been implemented on a CM2. We have also developed an $O(k \log N)$ algorithm for computing the visibility graph of a simple polygon, and have presented experimental results of implementing the algorithm on both the CM2 and the Maspar. Here, N is the length of a contour and k is the so-called link diameter of the polygon represented by the contour—the maximum number of straight line segments needed to connect any two points in the polygon.

7 Other topics

7.1 Robot hand/eye coordination [6, 53]

Traditional approaches to robot hand/eye coordination require that various components of the system be calibrated with respect to a common reference, but calibration is difficult and error-prone and may invalidate the complex, high-precision inverse kinematic computations that are also a feature of these approaches. We have developed a fundmentally new control technique that does not require any calibration and closely integrates visual feedback into the control mechanism. This is made possible by the introduction of a mapping, called the *Perceptual Kinematic Map*, from the control space of the manipulator directly onto a space defined by a set of *measurable* image parameters. Our strategy achieves robustness by monitoring qualitative rather than quantitative changes as it explores the surface defined by this mapping. Furthermore, it employs a Kalman-Bucy filter for additional robustness in measuring image parameters.

This approach can be applied to dynamic robot hand positioning tasks, such as catching, hitting, interception, etc., that involve an object (target) moving in the vicinity of the robot. We have examined the different levels at which visual input is involved in pursuing the dynamically-defined goal. A given task is transformed into one of constrained trajectory planning on the *perceptual control surface* defined by the PKM.

We have successfully demonstrated the feasibility of our approach by implementing the individual elements of the hand/eye system on available hardware—a 5-DOF Mitsubishi Movemaster II manipulator, a CCD camera, and a Macintosh II for processing and control. In particular, we have developed a module for the visual tracking of the hand using a Kalman filter. Although neither the camera nor the manipulator was calibrated—their relative position and orientation were unknown—our experiments indicated that the trajectories of the parameters we chose agreed closely with the predicted ones, and the mapping (PKM) was quite smooth.

7.2 Aerial image analysis [37, 46]

We have developed a straight line extractor that produces high quality line descriptions from aerial images.

The input to the line extractor is in the form of an edge image, where the contrast and direction of each edge pixel is specified. The system first scans the edge image left to right, top to bottom and assigns a line label for each scanned edge pixel, thereby generating a *label* image. At the end of this process each edge pixel has a line label associated with it and edge pixels that belong to the same line will be assigned the same line label. Also, with each line label, a record that stores the end points, the average contrast and the pixel support of the line is generated. The label image is used as a spatial index to further link fragmented lines. We have also developed techniques for eliminating many of the physically insignificant lines from the database, given that the domain of interpretation is aerial images, dominated by man-made objects.

A system for the detection of a class of buildings in aerial images has also been developed. The process of building detection is carried out in a hierarchical manner. Pairs of line segments are grouped to form *vertices*. Vertices are then grouped to form *edges*. Edges are composed into *edge-rings*. The process of building detection is posed as a graph search problem, where the nodes of the graph are *edges* and the links are connectivity relations. Closed edge-rings in this graph may correspond to roofs. Shadow analysis on *closed-rings* is used to generate building hypotheses. Some edge-rings are not fully closed. Heuristics based on the knowledge of the shapes of roofs are used to hypothesize edges top-down to close these edge-rings. A dynamic search scheme based on an Assumption based Truth Maintenance System (ATMS) helps in integrating bottom-up and top-down searches. The ATMS assists in enforcing constraints on the shapes of roofs extracted and in dealing with belief revisions associated with incremental additions, deletions and confirmations of edge hypotheses.

In Section 1 we described a search-based system for recognizing "objects" having variable numbers of parts which have only loosely constrained shapes; as mentioned there, this system has been successfully applied to finding cloverleaf intersections in aerial images of road networks.

7.3 Image matching and registration [16, 59]

A standard technique in image matching is to apply relaxation techniques to establish correspondences between patterns of localizable (point-like) features in the two images. We have extended point-pattern matching relaxation techniques to allow matching of both point-like and linear features. Our approach makes use of a compatibility function that relies on relative orientation information, which is translation and rotation invariant and which can be more reliably extracted from noisy images than can positional information. The function has been used to generalize the standard relaxation matching technique based on point patterns. It has been successfully applied to object recognition in synthetic aperture radar (SAR) imagery.

Another feature-based image matching method has been applied, with excellent results, to the registration of pairs of partially overlapping images in which large amounts of rotation and scaling have occurred between the two images and the images are devoid of significant features. An illuminant direction estimation method is first used to obtain an initial estimate of camera rotation. A small number of feature points are then located based on a Gabor wavelet model for detecting local curvature discontinuities. An initial estimate of scale and translation is obtained by pairwise matching of the feature points detected in two images. Finally, hierarchical feature matching is performed to obtain an accurate estimate of translation, rotation and scale. Experiments with synthetic and real images show that this algorithm yields accurate results when the scales of the pair of images differ by up to 10%, their overlap is as small as 35%, and the camera rotation between the two images is significant. Further details can be found in a paper in these proceedings.

7.4 Discrete and fuzzy geometry [10, 11, 29, 45]

In dealing with digital images in two and three dimensions, a rigorous understanding of geometric properties of such images is important for establishing the correctness of geometric algorithms. We have continued to investigate discrete geometric structures on various levels. Two contributions to this area during the past year include the study of polygonal arcs and polygons in three dimensions—in particular, their compact representation and the determination of their topological properties, particularly knottedness and linkedness; and the study of connectedness in three-dimensional digital images, leading to a proof of the long unresolved conjecture that 3D connectedness is not recognizable by a finite-state array automaton.

Geometrical properties of "objects" in unsegmented (gray level) images can be defined by regarding the objects as fuzzy subsets of the image. The study of geometric properties of fuzzy sets has begun to attract attention in the fuzzy set community. During the past year we have made two contributions to this area: we have developed a new method of defining the medial axis of a fuzzy set as a set of fuzzy disks whose sup is the set; and we have developed optimal algorithms for computing connectedness properties of fuzzy graphs (e.g., of graphs representing quantitative relations between pairs of objects).

References

[1] Daniel DeMenthon and Larry S. Davis, "Vision-Based Navigation and Parallel Computing—Second Anual Report", DACA76-88-C-0008, August 1990.

[2] R. Jasinschi, A. Rosenfeld, and K. Sumi, "The Perception of Visual Motion Coherence and Transparency: a Statistical Model", CAR-TR-512, AFOSR-86-0092, August 1990.

[3] Rae-Hong Park and Peter Meer, "Multiresolution Adaptive Least Squares Smoothing of Images", CAR-TR-513, AFOSR-86-0092, August 1990.

[4] Minas Spetsakis, "Linear Algorithm for the Point and Line Correspondence Problem", CAR-TR-514, DACA76-89-C-0019, September 1990.

[5] Azriel Rosenfeld, "Robust Image Understanding—Techniques and Applications—First Annual Report", DACA76-89-C-0019, September 1990.

[6] Jean-Yves Hervé, Rajeev Sharma and Peter Cucka, "Qualitative Coordination of a Robot Hand/Eye System", CAR-TR-516, DACA76-89-C-0019, October 1990.

[7] Tapio Seppänen and Vesa Vuohtoniemii, "Speedup Analysis of a Program for the Generalized Hough Transform on Two Different Multiprocessors", CAR-TR-517, DACA76-88-C-0008, October 1990.

[8] Sunil Arya and Daniel DeMenthon, "Textural Analysis of Range Images", CAR-TR-518, DACA76-88-C-0008, October 1990.

[9] Allen C. Sher and Azriel Rosenfeld, "Pyramid Cluster Detection and Delineation by Consensus", CAR-TR-519, AFOSR-86-0092, October 1990.

[10] Prabir Bhattacharya and Azriel Rosenfeld, "Polygonal Arcs and Polygons in Three Dimensions", CAR-TR-520, AFOSR-86-0092, October 1990.

[11] Sankar K. Pal and Azriel Rosenfeld, "A Fuzzy Medial Axis Transformation Based on Fuzzy Disks", CAR-TR-524, AFOSR-86-0092, November 1990.

[12] Doron Mintz, Peter Meer and Azriel Rosenfeld, "Consensus by Decomposition: A Paradigm for Fast High Breakdown Point Robust Estimation", CAR-TR-525, AFOSR-86-0092, December 1990.

[13] Rae-Hong Park and Peter Meer, "Edge-Preserving Artifact-Free Smoothing with Image Pyramids", CAR-TR-527, AFOSR-86-0092, December 1990.

[14] Zygmunt Pizlo and Azriel Rosenfeld, "Recognition of Planar Shapes from Perspective Images Using Contour-Based Invariants", CAR-TR-528, DACA76-88-C-0008, December 1990.

[15] Azriel Rosenfeld, "Image Analysis and Computer Vision: 1990", CAR-TR-529, AFOSR-86-0092, January 1991.

[16] Peter Cucka and Azriel Rosenfeld, "Linear Feature Compatibility for Pattern-Matching Relaxation", CAR-TR-530, DACA76-89-C-0019, January 1991.

[17] John (Yiannis) Aloimonos and Azriel Rosenfeld, "Visual Recovery", CAR-TR-531, DACA76-88-C-0008, January 1991.

[18] Ansel Y. Teng, Daniel DeMenthon and Larry S. Davis, "Stealth Terrain Navigation", CAR-TR-532, DACA76-88-C-0008, January 1991.

[19] Amit Bandopadhay and John (Yiannis) Aloimonos, "Image Motion Estimation by Clustering", CAR-TR-533, DACA76-89-C-0019 and IRI-90-57934, January 1991.

[20] Rajeev Sharma and John Aloimonos, "Robust Detection of Independent Motion: An Active and Purposive Solution", CAR-TR-534, DACA76-88-C-0008 and IRI-90-57934, January 1991.

[21] Liuqing Huang and Yiannis Aloimonos, "Relative Depth from Motion Using Normal Flow: An Active and Purposive Solution", CAR-TR-535, DACA76-89-C-0019 and IRI-90-57934, February 1991.

[22] P.J.Narayanan and Larry S. Davis, "Replicated Data Algorithms in Image Processing", CAR-TR-536, DACA76-88-C-0008, February 1991.

[23] Isaac Weiss, "Noise-Resistant Invariants of Curves", CAR-TR-537, N00014-91-J-1222, February 1991.

[24] Jun Ohya, Larry S. Davis and Daniel DeMenthon, "Recognizing 3-D Objects from Range Images and Finding Their Six Pose Parameters", CAR-TR-538, DACA76-88-C-0008, February 1991.

[25] Philip David, David Harwood and Larry Davis, "A Probabilistic, Local Feature, Stereo Correspondence Algorithm", CAR-TR-540, DACA76-88-C-0008, March 1991.

[26] Ling Tony Chen and Clyde P. Kruskal, "An EREW Parallel Algorithm for List Ranking Image Curves in $O(\log N)$ Time", CAR-TR-541, DAC76-88-C-0008, March 1991.

[27] Mohamed Abdel-Mottaleb and Azriel Rosenfeld, "Inexact Bayesian Estimation", CAR-TR-542, DACA76-89-C-0019, March 1991.

[28] A. Rangarajan and R. Chellappa, "Markov Random Fields and Neural Networks with Applications to Early Vision Problems", CAR-TR-543, AFOSR-90-0133, June 1991.

[29] Saibal Banerjee, "An Optimal Algorithm to find the Degrees of Connectedness in an Undirected Edge-weighted Graph", CAR-TR-544, DACA76-89-C-0019, March 1991.

[30] Isaac Weiss, "High Order Differentiation Filters That Work", CAR-TR-545, N00014-91-J-1222, March 1991.

[31] Doron Mintz, "Robust Consensus Based Edge Detection", CAR-TR-546, DACA76-89-C-0019, March 1991.

[32] R.S. Jasinschi, A. Rosenfeld, P. Cucka and K. Sumi, "Discriminating 3D Texture Patterns: The Velocity Histogram Method", CAR-TR-547, DACA76-89-C-0019, March 1991.

[33] Qinfen Zeng and Rama Chellappa, "Estimation of Illuminant Direction, Albedo, and Shape from Shading", CAR-TR-549, MIP-84-51010, June 1991.

[34] Jacob Beck, "Visual Processing in Texture Segregation", CAR-TR-554, AFOSR-88-0323, April 1991.

[35] E. Rignot and R. Chellappa, "Segmentation of Synthetic Aperture Radar Complex Data", CAR-TR-555, June 1991.

[36] V. Venkateswar and R. Chellappa, "Hierarchical Stereo Matching Using Feature Groupings", CAR-TR-556, F-49620-88-C-0067, May 1991.

[37] V. Venkateswar and R. Chellappa, "Extraction of Straight Lines in Aerial Images", CAR-TR-557, F-49620-88-C-0067, June 1991.

[38] Gem-Sun Young and Rama Chellappa, "Statistical Analysis of Inherent Ambiguities in Recovering 3-D motion from a Noisy Flow Field", CAR-TR-558, IRI-87-13585, July 1991.

[39] Ling Tony Chen and Larry S. Davis, "A Parallel Algorithm for the Visibility of a Simple Polygon Using Scan Operations", CAR-TR-559, DACA76-89-C-0019, June 1991.

[40] Zoran Durić and Yiannis Aloimonos, "Passive Navigation: An Active and Purposive Solution", CAR-TR-560, DACA76-89-C-0019 and IRI-90-57934, June 1991.

[41] E. Puppo, L. Davis, D. DeMenthon and Y.A. Teng, "Parallel Terrain Triangulation Using the Connection Machine", CAR-TR-561, DACA76-89-C-0019, June 1991.

[42] John Canning, "Recognizing Objects with Variable Appearances—The VAPOR System", CAR-TR-563, AFOSR-91-0239, August 1991.

[43] Cornelia Fermüller and Yiannis Aloimonos, "Estimating 3-D Motion from Image Gradients", CAR-TR-564, DACA76-89-C-0019, June 1991.

[44] Philippe Burlina, Daniel DeMenthon and Larry S. Davis, "Navigation with Uncertainty: I. Reaching a Goal in a High Collision Risk Region", CAR-TR-565, DACA76-89-C-0019, June 1991.

[45] Akira Nakamura and Azriel Rosenfeld, "Three-Dimensional Connected Pictures Are Not Recognizable By Finite-State Acceptors", CAR-TR-566, June 1991.

[46] V. Venkateswar and R. Chellappa, "A Hierarchial Approach to Detection of Buildings in Aerial Images", CAR-TR-567, F-49620-88-C-0067, August 1991.

[47] Philippe Burlina, Daniel DeMenthon and Larry S. Davis, "Navigation with Uncertainty: II. Avoiding High Collision Risk Regions", CAR-TR-568, DACA76-89-C-0019, July 1991.

[48] Rajeev Sharma, David M. Mount and Yiannis Aloimonos, "Probabilistic Analysis of Some Navigation Strategies in a Dynamic Environment", CAR-TR-569, DACA76-89-C-0019 and IRI-90-57934, July 1991.

[49] Saibal Banerjee and Azriel Rosenfeld, "Model-Based Cluster Analysis", CAR-TR-570, AFOSR-91-0239, August 1991.

[50] Mohamed Abdel-Mottaleb and Azriel Rosenfeld, "'Qualitative' Bayesian Estimation of Digital Signals and Images", CAR-TR-571, DACA76-89-C-0019, August 1991.

[51] Sven Dickinson, "The Recovery and Recognition of Three-Dimensional Objects Using Part-Based Aspect Matching", CAR-TR-572, AFOSR-91-0239, August 1991.

[52] D.A Usikov, M.I. Kolesnik and I.N. Sitenko, "Three Dimensional Vision for Autonomous Vehicles (Algorithms, Problems, Perspectives)", CAR-TR-573, August 1991.

[53] Rajeev Sharma, Jean-Yves Hervé and Peter Cucka, "Analysis of Dynamic Hand Positioning Tasks Using Visual Feedback", CAR-TR-574, DACA76-89-C-0019, August 1991.

[54] Saibal Banerjee and Azriel Rosenfeld, "MAP Estimation of Context-Free Grammars", CAR-TR-575, AFOSR-91-0239, August 1991.

[55] Doron Mintz, "Robustness By Consensus", CAR-TR-576, AFOSR-91-0239, August 1991.

[56] Ansel Y. Teng, Daniel DeMenthon and Larry S. Davis, "Region-to-Region Visibility Analysis Using Massively Parallel Hypercube Machines", CAR-TR-578, DACA76-89-C-0019, September 1991.

[57] Ting-Hu Wu, Gem-Sun Young and Rama Chellappa, "A Simple Kinematic Model Based Approach for 3-D Motion and Structure Estimation", CAR-TR-579, N00014-89-5-1598, September 1991.

[58] Mohamed Abdel-Mottaleb and Azriel Rosenfeld, "Bayesian Estimation Based on Partial Information", CAR-TR-582, DACA76-89-C-0019, September 1991.

[59] Qinfen Zheng and Rama Chellappa, "A Computational Vision Approach to Image Registration", CAR-TR-583, DACA76-89-C-0019, September 1991.

Image Understanding Research at Rochester

Dana Ballard, Christopher Brown, Randal Nelson
Computer Science Department
University of Rochester
Rochester, NY 14607

Abstract

Work in animate vision continues at all levels from low-level, preattentive object tracking and segmentation utilities to Bayesian techniques and decision theory for controlling and reasoning about the vision process. We have had success with pose-invariant object recognition, which we pursued using color and projectively invariant geometric features. Learning is taking on a position of increasing importance in our work as we strive for more adaptive behavior that needs less a priori structuring.

1 Laboratory and Aims

Our laboratory continues to expand to meet our needs for more sophisticated analysis, data-gathering, and control hardware for our work in animate vision [Ballard 1991a,b]. The laboratory has been enhanced by a DataGlove input device for obtaining input from humans for a variety of recognition projects and Utah hand training projects. Recognition strategies for dynamic trajectories in various spaces (angle spaces, Cartesian space) we are investigating include hidden Markov models and recurrent networks [Cortes et al; Simard 1991; Simard et al. 1990a,b; 1991a,b]. The dataglove can also be worn by the Utah hand, and self-calibration is being investigated.

A bidirectional communications library for the Puma robot (ROBOCOM) was written by Brian Yamauchi, John Soong, Tim Becker, and Prakash Das for use in the Juggler [Yamauchi 1991], Checkers [Marsh et al. 1991], and open-loop gaze stabilization projects [Soong and Brown 1991]. ROBOCOM is much faster than the earlier BOTLIB package since it does not use the multi-layered ISO-standard structure for communication. Robot head positions can be read back into a host computer at between 15 and 40 Hz., depending on how much other computation is being done by VAL.

As more intimate connection is made between framerate computations in special hardware and the memory of general purpose computers, the visual computations that can be performed in real time grow more complex. We have expanded our real-time computing resources with two more MIMD multi-computers. The Silicon Graphics computer is a modern resource with eight 40 MHz MIPS chips. We plan to use its graphics capabilities to produce realistic scenes for computer analysis and for input to psychophysical experiments. the Transputer array consists of eight 4MByte T805 TRAMS and a smart VME interface that lets the Transputers directly read and write VME memory and hence control the MaxVideo and hand. The interface also has serial ports that allow robot control and dataglove input. The KiwiVision product will allow Transputers to access the MaxBus directly, speeding higher-level visual processing. A Sun4/330 workstation acts as a host terminal system.

2 Low-level Gaze Control and Segmentation

We use cooperating low-level gaze controls to improve tracking and segmentation performance. We have successfully combined vergence and tracking capabilities. Vergence [Olson and Coombs 1990] limits the amount of visual information coming from a scene, and can augment other segmentation techniques. If only features having zero disparity are passed through a filter [von Kaenel 1991], then vergence implements a disparity-based filter that restricts attention to the "horopter", or 3-D locus of points that have zero disparity at the current vergence. Thus disparity filtering can separate a spatially coherent target from foreground and background distractors. Used with a fovea or window filter that limits visual information to a particular solid angle, disparity filtering limits visual information to a small volume of space. We use the horopter and foveal constraints in a tracking system that pursues a moving object through a field of distractors [Coombs and Brown, 1990, 1991; Coombs et al. 1990].

Object tracking continues to be an important area of our work [Bandyopadhyay and Ballard 1991]. We have successfully used $\alpha-\beta-\gamma$ filters as predictors to improve the performance of visual tracking loops that move the camera to keep a moving object centered. We have studied various approaches to the problem of delay in control loops [Brown and Coombs 1991]. We have also implemented open-loop gaze stabilization capabilities that keep the camera pointed at a 3-space point despite arbitrary head motion by making compensating pans and tilts. The crux of the computation is an inverse kinematic model of the robot head, and we have also derived the inverse Jacobian so we can use a position or velocity formulation [Soong and Brown 1991]. The open-loop

stabilization capability has been extended to maintain the gaze on a point moving along a known trajectory.

3 Visual Attention

Eyemovements are an important aspect of animate vision [Ballard 1991]. Augmented hidden Markov models (AHMMs) can learn graph structures representing either action sequences, adjacency and connectivity information about objects, or even control structures of observed algorithms [Rimey and Brown 1990, 1991a,b,c] The models are augmented in the sense that they can, at run-time, modify their output based on scene information. Their learning capabilities allow them to adapt to slowly-varying scene characteristics. They are used in a generative mode to output learned behavior. Behavior can be learned either in a "where" mode, in which (say) specific visual locations are foveated in sequence, or in a "what" mode, which sequentially foveates a desired sequence of features in the image, regardless of their location. The training can come from an instructor, or from another program that is producing the behavior as the result of a cognitive process. The latter case is like learning a skill, in that a visuo-motor skill structure can be developed that does not have to repeat the reasoning process, and thus can run more efficiently.

Using the augmentation feature of the hidden Markov model, the output of the "where" and "what" systems each can be fed back to the other system, in order to train the system to output a sequence of desired features or objects in their expected locations in the image. The system also incorporates the adaptive visual feedback cues and a control scheme for verifying expectations using foveal image data.

The what-where-AHMM has a "what" part, a "where" part, and an output combiner. The what-part contains two stages. The first stage is an AHMM whose output symbols F_t, called *what-symbols*, are feature vectors intended to describe an object or characteristics of objects. Such feature vectors are assumed to have been computed for each pixel in the (peripheral) image. The second stage of the what-part performs a "what-to-where" mapping, meaning that it maps a feature vector into the set $\mathbf{G_t}$ of camera movement commands, called *where-symbols*, that would cause those locations to be foveated (centered in the image).

If each $\mathbf{G_t}$ contains exactly one element, the output sequence will fixate the desired objects in the scene. Each $\mathbf{G_t}$ does not generally contain exactly one element, so some method must be developed to select among the choices. One option is to use a where-AHMM to help pick among the choices. In fact, the what-AHMM can be made to help the where-AHMM with its *own* choices.

The where-part contains two stages, similar to those in the what-part. First it has an AHMM, which outputs a sequence of where-symbols O_t. Secondly it has a "where-to-what" mapping that determines for each where-symbol O_t the location in the current (peripheral) image it corresponds to, and outputs the set of feature vectors $\mathbf{P_t}$ in that local area of the image. The AHMM in the where-part uses as feedback a sequence of sets of where-symbols $\mathbf{G_t}$, which is the output of the what-part.

Finally, the output combiner determines the overall output of the what-where-AHMM. The overall output at time step t is Z_t, a where-symbol (i.e. a camera movement command), selected as the element of the set $\mathbf{G_t}$ that has the smallest distance to the symbol O_t.

The what-where-AHMM operates as follows. At each time step, each of the two parts produces a set of feedback symbols that reflects its own preference for action. Each then updates its own preferences taking the other's into account, and then generates its own final preference for action at that time step. The set of final preferences is reduced to a single output symbol by the output combiner.

4 Motion Detection and Recognition

Motion analysis is an important process in animate vision systems. In particular, a system that interacts visually with a dynamic world must necessarily make decisions about moving objects. Recent work has addressed the problem of detecting independently moving objects from a moving sensor [Nelson 1991], and identifying the source of the motion once it has been detected [Polana and Nelson 1991].

Two complementary methods for the detection of moving objects by a moving observer have been developed. The first is based on the fact that, in a rigid environment, the projected velocity at any point in the image is constrained to lie on a 1-D locus in velocity space whose parameters depend only on the observer motion. If the observer motion is known, an independently moving object can, in principle, be detected because its projected velocity is unlikely to fall on this locus. This principle was adapted to use partial information about the motion field and observer motion that can be rapidly computed from real image sequences (e.g. gradient parallel image flow). The method was implemented to run in real time (approximately 1/15 second latency and update) using parallel pipelined MAXVIDEO image processing hardware. The second method utilizes the fact that the apparent motion of a fixed point due to smooth observer motion changes slowly, while the apparent motion of many moving objects such as animals or maneuvering vehicles may change rapidly. The motion field at a given time can thus be used to place constraints on the future motion field which, if violated, indicate the presence of an autonomously maneuvering object. In both cases, the qualitative nature of the constraints allows the methods to be used with the inexact motion information typically available from real image sequences. This was also implemented to run in real time on our parallel processing testbed.

The goal of the ongoing research in motion recognition is to demonstrate that robustly computable motion features can be used directly as a means of recognition. This contrasts with the more traditional approach to motion analysis, which has emphasized instead, a reconstructive approach. Specifically, the goal is to design, implement, and test a general framework for recognizing both distributed motion activity on the basis of *temporal texture*, and complexly moving, compact objects on the basis of their *action* . This *recognition* approach contrasts with

the *reconstructive* approach that has typified most prior work on motion. The underlying motivation is the observation that, for objects that typically move, it is frequently easier to identify them when they are moving than when they are stationary. Specifically, in the case of temporal texture, we extract statistical spatial and temporal features from approximations to the motion field and use techniques analogous to those developed for gray-scale texture analysis to classify regional activities such as windblown trees, ripples on water, or chaotic fluid flow, that are characterized by complex, non-rigid motion. For action identification, the spatial and temporal arrangement of motion features can be used in conjunction with simple geometric image analysis to identify complexly moving objects such as machinery and locomoting people and animals. This work has practical applications in monitoring and surveillance, and as a component of a sophisticated visual system.

Current research has concentrated on temporal texture identification. Temporal textures represent movements that are characterized by some degree of spatial invariance. Such motion is commonplace in the natural world. Examples include ripples on a pool, leaves or grass blowing in the wind, turbulence in cloud patterns, and the behavior of a flock of birds or a crowd of people. The basic approach rests on the computation of robust motion features. Some examples are differential quantities such as divergence and curl, motion discontinuities of various types, and temporal frequency information. The variety of possible features is much greater than for a gray-scale image, both because the motion field is a vector rather than a scalar array, and because temporal as well as spatial relations can be considered.

The main challenge (as in gray-level texture classification) is to come up with a generally useful set of regional motion features that can be used for recognition. Various invariances, e.g. to scale, rotation or temporal compression may or may not be desirable, depending on the application. We have acquired and digitized a motion database containing a number of real-world temporal textures, and identified a number of useful features. Most of these are based on the normal flow, which is easier to obtain than the full motion field.

The notion of temporal texture is analogous to that employed in statistical methods of gray-level texture classification: we want to compute temporal features that can be used to characterize a viewing region filled by the texture. These features should at least be invariant under translation of the texture with respect to the viewing region. In some cases, invariance with respect to rotation, illumination, and spatial and temporal scaling is also important. We have investigated a number of features based on local properties of the normal flow field. In general, all these features involve spatial and sometimes temporal averaging of local properties. Useful feature include spatial and temporal uniformity measures, local curl and divergence, and cooccurence statistics on the direction histogram. A small set of these allows accurate classification of a number of temporal textures.

5 Color for Robust Recognition and Location

Color histograms can be robust to object obscuration, pose, or configuration (for nonrigid objects) [Swain 1990; Swain et al. 1989, 1990a,b]. The algorithms described here were implemented and run at approximately 5Hz to recognize objects reliably from a catalog of 75 rigid and nonrigid objects. Given a pair of histograms, I and M, each containing n buckets, the intersection of the histograms is defined to be

$$\sum_{j=1}^{n} \min(I_j, M_j)$$

The result of the intersection of a model histogram with an image histogram is the number of pixels from the model that have corresponding pixels of the same color in the image. To obtain a fractional match value between 0 and 1 the intersection is normalized by the number of pixels in the model histogram. The match value is then

$$\frac{\sum_{j=1}^{n} \min(I_j, M_j)}{\sum_{j=1}^{n} M_j}.$$

This basic matching algorithm runs in time linear in the number of items in the database. Ingenious use of selected features and a pre-indexing of the catalog of objects allows the matching to take place in constant time (for up to 75 objects).

The normalized histogram intersection match value is a metric in feature-space, and so can be used for nearest neighbor classification. It has the useful property that it is not reduced by distracting pixels in the background. The histogram intersection match value is only increased by a pixel in the background if

- the pixel has the same color as one of the colors in the model, and
- the number of pixels of that color in the object is less than the number of pixels of that color in the model.

Object location is possible with color histograms if we know their approximate sizes. A histogram back-projection algorithm answers the question "Where are the colors in the image that belong to the object being looked for (the *target*)?" The algorithm deemphasizes colors that appear in other objects having different histograms, so that they are less likely to distract the search mechanism. Experiments show that the technique works for objects in cluttered scenes under realistic conditions.

6 Projective Invariants

Although geometric features vary with the imaging geometry and hence viewpoint, invariant measures (such as moments), features (such as conics), and representations (such as invariant representations of planar curves) can be derived [Forsyth et al. 1990a,b; Brown 1991a,b]. Recent work in geometric (affine and projective) invariants shows promise of discovering feature-extraction processes for planar and three-dimensional features that

yield results not affected by perspective distortion. This work draws on the substantial tradition and techniques of projective geometry. This approach has certain technical difficulties, but is promising for model-based vision since it leads to pose-independent recognition. A DARPA-sponsored workshop (jointly funded by the European ESPRIT project) brought together a number of researchers from Europe and the US; the proceedings will appear as a book published by MIT Press.

7 Multi-model Parallel Programming

We have been studying and using parallel computation for many years [Crowl thesis, Quiroz thesis]. Our experience indicates that animate and integrated vision systems are inherently parallel and require multiple models of parallelism [Weems et al. 1991]. A system that integrates perception, planning, and acting will utilize a wide variety of data structures, algorithms, and models of processes and communication. One way to achieve this multiplicity is through heterogeneous hardware systems. However, it is often desirable to include many intimately-cooperating models of parallel computation (say programs written in different languages) in one computer.

The functional demands of integrated parallel computer vision systems within one multiprocessor imply the following technical requirements.

- Support is needed for to facilitate the integration of modules developed by different people at different times in different languages under different models of parallelism.
  - Easily understood, standard interaction mechanisms, such as shared memory and procedure call, are needed. On the other hand, one should have some choice in procedure calls — highly protected RPC-like mechanisms as well as efficient and fast calls should be supported.
  - Translation between model semantics is needed.
- Support is needed to allow the appropriate parallel computation models to function and communicate effectively.
  - A range of protection mechanisms is needed, tailored to the degree of trust between modules, allowing different levels of speed and safety to coexist within an application.
  - Shared software modules and active data structures should be easily created, incorporated, and removed dynamically.
  - The kernel and the user should share data, allowing for efficient user-level interactions with the system (for locks, interrupts, etc.) and system performance statistics gathered by the kernel. This facility is useful for time-critical applications as well.
  - There should be extensive scheduling capability in user space, both for the sake of performance and to allow applications to implement different kinds of processes.

An example of an integrated multi-model sensorimotor system is the Rochester Checkers player [Marsh et al. 1992, Karlsson 1991]. Four models of parallelism are used, each embodied in a corresponding language or library package. Uthread is a Presto-like parallel programming package that we call from C++ programs. Lynx is a message-passing language. The Uniform System is a data-parallel programming tool furnished by BBN-ACI. Multilisp is a parallel LISP. The Psyche operating system developed at Rochester provides the efficient multi-model substrate.

In our experience with Checkers, Psyche proved to be a very practical substrate. For example, the Lynx checkers-playing program and the uniform system image-analysis library were resurrected after several years of disuse and plugged in essentially unmodified. The serial robot communications interface was written in 1988 as an exercise and also was plugged in unmodified. The vision, move planning, board module, and move recognition modules, as well as necessary Psyche support for the particular models we used, were developed in parallel. Coding these modules was a part-time activity extending over several weeks. Integration was a full-time activity that only took a few days. At integration time, we made (and changed) many decisions about which modules should communicate directly with each other and which should use the indirect shared memory mechanism.

The Checkers program responds immediately to its competition. Generally, another aspect of parallel systems software for animate vision is that it should be real-time – that is, it should compute fast enough to survive. We feel that the current generation of real-time operating systems have a fatal flaw: the requirement that computational demands be known in advance. Given such knowledge, current real-time systems allow users to meet the anticipated deadlines and demands. But a true animate system will not have the luxury of a predictable universe. It will have to engage in "satisficing" behavior that chooses what to do, how far to pursue goals that are not working out, when to preempt current tasks to deal with emergencies, etc. This mixture of real-time AI, resource allocation decision making, and operating system design seems vital to the practical deployment of animate systems.

8 Vision and Task Learning

Learning systems are being built using several different techniques [Jain thesis, Simard thesis, Whitehead 1991, Wixson 1991, Rimey and Brown 1991]]. Current reinforcement learning techniques are only practical when the amount of state to be represented is small (less than 100 bits, say). One way to reduce the burden of representation to just that essential for the task is to use markers (temporary variables that record partial computational results). This notion of markers was introduced as a general method of object-centered computation The notion was that markers provided a local context to resolve reference ambiguity. In routine activity, long causal chains are not necessary. It turns out that the fixation point strategy can be thought of as a kind of marker that

has the right kind of transfer for learning many tasks.

A simulated block-stacking system at Rochester provides an example [Whitehead et al. 1990a,b, 1991; Whitehead 1991a,b,c; Ballard and Whitehead 1992] On each trial, the system is presented with a pile of colored blocks. A pile consists of any number of blocks arbitrarily arranged. Each block is uniformly colored either red, green, or blue. The system can manipulate the pile by picking and placing objects. When the system arranges the blocks into a successful configuration, it receives a positive reward and the trial ends. For example, one extremely simple block stacking task is for the system to learn to pick up a green block. In this case, the successful configurations consist just of those states where the system is holding a green object. The system learns to arrange arbitrary configurations of blocks into successful configurations. The key point here is that the marker encoding obviates the need for explicit coordinates.

We are also exploring the paradigm of visual object search, which uses prior knowledge about the likelihood of finding objects in the proximity of other objects and incorporates a learning component [Wixson and Ballard 1991; Wixson 1991]

9 Vision and Planning

We are currently considering the "Where to look next?" problem, and are applying Bayesian and planning techniques to the sequential decision-making aspects of cognitively directed visual activity. The work by Hartman [Hartman 1990, 1991a,b] addresses the difficult "cost of planning" problem and will be useful in a practical system in which difficult planning problems could lead to dysfunctional use of scarce resources of computation and time. Previous work in integrating low-level segmentation with high-level object recognition used a Bayesian approach and Markov random fields [Chou et al 1991, Swain 1990].

10 Related Theses, 1990-1991

Crowl, L.A., "Architectural adaptability in parallel programming," Ph.D. Thesis (and TR 381), Computer Science Dept. (Advisor: T.J. LeBlanc), May 1991.

To create a parallel program, programmers must decide what parallelism to exploit, and choose the associated data distribution and communication. Since a typical algorithm has much more potential parallelism than any single architecture can effectively exploit, programmers usually express only the exploitation of parallelism appropriate to a single machine. Unfortunately, parallel architectures vary widely. A program that executes efficiently on one architecture may execute badly, if at all, on another architecture. To port such a program to a new architecture, we must rewrite the program to remove any ineffective parallelism, to introduce any parallelism appropriate for the new machine, to redistribute data and processing, and to alter the form of communication. Architectural adaptability is the ease with which programmers can tune or port a program to a different architecture. The thesis of this dissertation is that control abstraction is fundamental to architectural adaptability for parallel programs. With control abstraction, we can define and use a rich variety of control constructs to represent an algorithm's potential parallelism. Since control abstraction separates the definition of a construct from its implementation, a construct may have several different implementations, each providing different exploitations of parallelism. By selecting an implementation for each use of a control construct with annotations, we can vary the parallelism we choose to exploit without otherwise changing the source code. We present Matroshka, a programming model that supports architectural adaptability in parallel programs through object-based data abstraction and closure-based control abstraction. Using the model, we develop several working example programs, and show that the example programs adapt well to different architectures. We also outline a programming method based on abstraction. To show the implementation feasibility of our approach, we describe a prototype language based on Matroshka, describe its implementation, and compare the performance of the prototype with existing programs.

Hartman, L., "Decision theory and the cost of planning," Ph.D. Thesis (and TR 355), Computer Science Dept. (Advisor: D.H. Ballard), September 1990.

This thesis shows how it is possible for a planner to make decisions that are sensitive to the computational resources that it expends. The main feature of the approach is to ignore the distinction between planning and executing and interpret planning procedures as actions with uncertain outcomes. A planner can then use standard decision theoretic techniques to select which among a set of alternative procedures is a better gamble. The hypotheses that underlie this work are that an autonomous agent must monitor its resource expenditure in order to be successful and that computation is an important and valuable resource. The main points of this thesis are: (1) it is possible for planners to make local inexpensive decisions that account for essentially their entire resource expenditure; (2) there are limitations to what a planner is able to infer about optimal strategies; and (3) it is possible for planners to be sensitive to the statistical properties of their problem-solving environments.

Jain, S., "Learning in the presence of additional information and inaccurate information," Ph.D. Thesis (and TR 357), Computer Science Dept. (Advisor: M.A. Fulk), September 1990.

Inductive inference machines (IIMs) model language and scientific learning. In the classical model, a machine attempts to construct an explanation about a phenomenon as it is receiving data about that phenomenon. The machine is said to be successful if it ultimately succeeds in explaining the phenomenon. This is a naive model of science. For one thing, a scientist has more information available than just the result of experiments. For example, a scientist may have some knowledge about the complexity of the phenomenon he (she) is trying to learn. For another, the result of the scientist's investigation need not be the final theory. Finally, a scientist may already have some approximate explanation of the phenomenon. The study of such additional information

constitutes the first part of this thesis. In the real world our input is rarely free of error. Inputs usually suffer from noise and missing data. The study of different notions of such inaccuracies in the input is the focus of the second part of this thesis.

Quiroz Gonzlez, C.A.,"Systematic detection of parallelism in ordinary programs" Ph.D. Thesis (and TR 351), Computer Science Dept. (Advisor: D. Baldwin), May 1991.

This dissertation discusses a general model for compilers that take imperative code written for sequential machines (ordinary code) and detect the parallelism in that code that is compatible with the semantics of the underlying programming language. This model is based on the idea of separating the concerns of parallelism detection and parallelism exploitation. This separation is made possible by having the detection component provide an explicit representation of the parallelism available in the original code. This explicitly parallel representation is based on a formalization of the notion of permissible execution sequences for a given mass of code. The model discussed here prescribes the structure of the parallelism detector. This structure depends on (1) recognizing a hierarchical structure on a graph representation of the program, and (2) separately encoding parallelization conditions and effects. Opportunities for parallelization can then be discovered by traversing the hierarchical structure from the bottom up. During this traversal, progressively larger parts of the program are compared against the independently encoded conditions, and transformed when the conditions are satisfied. The hierarchy guarantees that the results of transforming a piece of a program propagate in time to affect the possible parallelization of larger pieces. Although some of the algorithms used have exponential worst cases for general graphs, their observed behavior on real flow graphs is no worse than quadratic on the size of the original program.

Simard, P.Y., "Learning state space dynamics in recurrent networks," Ph.D. Thesis (and TR 383), Computer Science Dept. (Advisor: D.H. Ballard), June 1991.

Fully recurrent (asymmetrical) networks can be used to learn temporal trajectories. The network is unfolded in time, and backpropagation is used to train the weights. The presence of recurrent connections creates internal states in the system which vary as a function of time. The resulting dynamics can provide interesting additional computing power but learning is made more difficult by the existence of internal memories. This study first exhibits the properties of recurrent networks in terms of convergence when the internal states of the system are unknown. A new energy functional is provided to change the weights of the units in order to control the stability of the fixed points of the network's dynamics. The power of the resultant algorithm is illustrated with the simulation of a content addressable memory. Next, the more general case of time trajectories on a recurrent network is studied. An application is proposed in which trajectories are generated to draw letters as a function of an input. In another application of recurrent systems, a neural network exhibits certain temporal properties observed in human callosally sectioned brains. Finally the proposed algorithm for stabilizing dynamics around fixed points is extended to one for stabilizing dynamics around time trajectories. Its effects are illustrated on a network that generates Lisajous curves.

Swain, M.J., "Color indexing," Ph.D. Thesis (and TR 360), Computer Science Dept. (Advisor: D.H. Ballard), November 1990.

Computer vision is embracing a new research focus in which the aim is to develop visual skills for robots that allow them to interact with a dynamic, realistic environment. To achieve this aim, new kinds of vision algorithms need to be developed that run in real time and subserve the robot's goals. Two fundamental goals are determining the identity of an object with a known location and determining the location of a known object. Color can be successfully used for both tasks. This dissertation demonstrates that color histograms of multicolored objects provide a robust, efficient cue for indexing into a large database of models. It shows that color histograms are stable object representations in the presence of occlusion and over change in view, and that they can differentiate among a large number of objects. For solving the identification problem, it introduces a technique called histogram intersection, which matches model and image histograms, and a fast incremental version of histogram intersection which allows real-time indexing into a large database of stored models. It demonstrates techniques for dealing with crowded scenes and with models with similar color signatures. For solving the location problem it introduces an algorithm called histogram backprojection, which performs this task efficiently in crowded scenes.

DARPA-Related Publications
University of Rochester
Computer Science Department

July 1990 - June 1991

Ballard, D.H., "Animate vision," *Artificial Intelligence Journal 48*, 57-86, 1991.

Ballard, D.H., "Animate vision: An evolutionary step in computational vision," *J. of the Inst. of Electronics, Information, and Communication Engineers 74*, 4, 343-348, April 1991.

Ballard, D.H., "Models of human intelligence: A sub-symbolic approach," invited manuscript, Wolfson College Lecture Series, Wolfson College, Oxford, January 1991.

Bandopadhay, A. and D.H. Ballard, "Egomotion perception using visual tracking," *Computational Intelligence 7*, February 1991.

Bogucz, K.J., "Document image segmentation via top-down projection profile and run-length analysis," Bellcore Technical Memorandum TR-ARH-018037, August 1990.

Brown, C.M., "An empirical investigation of differential invariants," *DARPA/Esprit Workshop*, Reykjavik, March 1991.

Brown, C.M. and R.C. Nelson, "Image understanding research at Rochester," *Proc., DARPA Image Understanding Workshop*, Pittsburgh, PA, September 1990.

Coombs, D.J. and C.M. Brown, "Intelligent gaze control in binocular vision," *Proc., IEEE Workshop on Intelligent Control,"* Philadelphia, PA, September 1990.

Coombs, D.J. and C.M. Brown, "Cooperative gaze-holding in binocular vision," *IEEE Control Systems*, June 1991.

Coombs, D.J., T.J. Olson, and C.M. Brown, "Gaze control and segmentation," *Proc., AAAI Qualitative Vision Workshop*, Boston, MA, August 1990; *1990-91 Computer Science and Engineering Research Review*, Computer Science Dept., U. Rochester, 7-11, September 1990.

Forsyth, D., J.L. Mundy, A. Zisserman, and C.M. Brown, "Invariance, a new framework for vision," *Int'l. Conf. on Computer Vision*, Osaka, Japan, December 1990.

Forsyth, D., J.L. Mundy, A. Zisserman, and C.M. Brown, "Projectively invariant representations using implicit algebraic-curves," *Springer-Verlag Lecture Notes in Computer Science*, 427-436, 1990.

Hartman, L.B., "A decision theoretic approach to controlling the cost of planning," *Proc., 3rd Int'l. Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, January 1991.

Hartman, L.B., "Decision theory and the cost of planning," Ph.D. Thesis and TR 355, Computer Science Dept., U. Rochester, September 1990.

Hartman, L.B., "Uncertainty and the cost of planning," TR 372, Computer Science Dept., U. Rochester, February 1991.

Karlsson, J., "A motion planner for Checkers," *Proc., 6th Annual U. Buffalo Graduate Conf. on Computer Science*, Buffalo, NY, March 1991.

Marsh, B.D., C.M. Brown, T.J. LeBlanc, M.L. Scott, T.G. Becker, P.Ch. Das, J. Karlsson, and C.A. Quiroz, "The Rochester checkers player: Multi-model parallel programming for animate vision," TR 374, Computer Science Dept., U. Rochester, June 1991.

Nelson, R.C., "Qualitative detection of motion by a moving observer," *Proc., AAAI-90 Workshop on Qualitative Vision*, Boston, MA, July 1990; *Proc., DARPA Image Understanding Workshop*, Pittsburgh, PA, September 1990; *Proc., IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, Hawaii, June 1991.

Olson, T.J. and D.J. Coombs, "Real-time vergence control for binocular robots," *Proc., DARPA Image Understanding Workshop*, September 1990.

Rimey, R.D. and C.M. Brown, "HMMs and vision: Representing structure and sequences for active vision using hidden Markov models," TR 366, Computer Science Dept., U. Rochester, January 1991.

Rimey, R.D. and C.M. Brown, "Selective attention as sequential behavior: Modeling eye movements with an augmented hidden Markov model," *Proc., DARPA Image Understanding Workshop*, September 1990.

Rimey, R.D. and C.M. Brown, "Sequences, structure, and active vision," *Proc., IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, Hawaii, June 1991.

Shaffer, C.A., H. Samet, and R.C. Nelson, "QUILT: A geographic information system based on quadtrees," *Int'l. J. of Geographic Inf. Systems 4*, 2, 103-131, August 1990.

Simard, P.Y., "Learning state space dynamics in recurrent networks," TR 383 and Ph.D. Thesis, Computer Science Dept., U. Rochester, June 1991.

Simard, P.Y., C. Cortes, and B. Victorri, "An energy function for changing the dynamics around state trajectories in recurrent networks," presented, *Snowbird Conf.*, Snowbird, UT, April 1991.

Simard, P.Y. and Y. LeCun, "Trajectory generation using the 'reverse' TDNN architecture," presented, *Snowbird Conf.*, Snowbird, UT, April 1991.

Simard, P.Y. and G.E. Mailloux, "Vector field restoration by the method of convex projections," *Computer Vision Graphics, and Image Processing 52*, 3, December 1990.

Simard, P.Y., J.P. Raysz, and B. Victorri, "Shaping the state space landscape in recurrent networks," presented, *IEEE Conf. on Advances in Neural Information Processing Systems 3* (NIPS), 105-112, December 1990.

Swain, M.J., "Color Indexing," TR 360 and Ph.D. Thesis, Computer Science Dept., U. Rochester, November 1990.

Swain, M.J., "Parameter learning for Markov random fields with highest confidence first estimation," TR 350, Computer Science Dept., U. Rochester, August 1990.

Swain, M.J. and D.H. Ballard, "Color indexing," *1990-91 Computer Science and Engineering Research Review*, Computer Science Dept., U. Rochester, 12-13, September 1990.

Swain, M.J. and D.H. Ballard, "Indexing via color histograms," *Proc., DARPA Image Understanding Workshop*, September 1990; *Proc., Int'l. Conf. on Computer Vision* (ICCV 90), Kyoto, Japan, December 1990.

Swain, M.J., L.E. Wixson, and P.B. Chou, "Efficient parallel estimation for Markov random fields," in M. Henrion, R. Schacter, L.N. Kanal, and J. Lemmer (Eds.). *Uncertainty in Artificial Intelligence: Volume V (Proc., 5th Workshop on Uncertainty and Artificial Intelligence*, Windsor, Ontario, August 1989). New York: Elsevier Science Pub. Co., 407-422, 1990.

Whitehead, S.D., "Complexity and cooperation in reinforcement learning," *8th Machine Learning Workshop*, Evanston, IL, June 1991.

Whitehead, S.D., "A framework for integrating perception, action, and trial-and-error learning," *Proc., AAAI Spring Symp. on Integrated Architectures*, Palo Alto, CA, March 1991.

Whitehead, S.D., "A study of cooperative mechanisms for faster reinforcement learning," TR 365, Computer Science Dept., U. Rochester, March 1991.

Whitehead, S.D. and D.H. Ballard, "Active perception and reinforcement learning," *Neural Computation 2*, 4, 409-419, 1990.

Whitehead, S.D. and D.H. Ballard, "Learning to perceive and act by trial and error," *Machine Learning 7*, 1, 45-83, 1991.

Whitehead, S.D., R.S. Sutton, and D.H. Ballard, "Recent advances in reinforcement learning and their implications for intelligent control," *Proc., IEEE Int'l. Symp. on Intelligent Control*, 1990.

Wixson, L.E., "Scaling reinforcement learning techniques via modularity," *Proc., 8th Int'l. Workshop on Machine Learning*, Evanston, IL, June 1991.

Wixson, L.E., and D.H. Ballard, "Real-time qualitative detection of multi-colored objects for object search," *Proc., AAAI Workshop on Qualitative Vision*, Boston, MA, July 1990.

Yamauchi, B. and R.C. Nelson, "A behavior-based architecture for robots using real-time vision," *Proc., IEEE Int'l. Conf. on Robotics and Automation*, Sacramento, CA, April 1991.

Image Understanding Research at GE

J.L. Mundy*
Box 8
G.E. Corporate Research and Development
Schenectady, NY 12309

Abstract

Recent progress in image understanding research at GE is described. The focus of GE's program in IU is on the application of geometric constraint models and geometric invariants to the recognition and representation of objects. Progress on the development of various object-oriented software environments to support IU research and applications are also described.

1 Overview

1.1 An Emphasis on Geometry

Image understanding research and applications at GE have been centered around geometric descriptions and geometric reasoning for representing and recognizing objects. We have developed this geometric theme over the past decade with emphasis on object recognition and associated approaches for representing objects to facilitate recognition.

Our current work is developing along a number of related geometric themes.

1.1.1 Constraint-Based Modeling

The conventional approaches to object recognition have been based on fixed polyhedral models or models with fixed relationships between components. We have been developing a system for representing broad categories of objects by defining an object in terms of geometric relations, or constraints. Any specific object of the class is considered to be a solution of the constraint system. In most cases, there is a continuum of solutions so a specific object is selected which satisfies the constraints as well as optimality criteria derived from image features.

We have applied constraint-based modeling to the problem of constructing site models from aerial images in support of image simulation for mission training and rehearsal and site models to support image intelligence applications, a goal of the RADIUS program.

1.1.2 Geometric Invariants

A significant body of results on the construction of geometric invariants to projective and affine transformations have been developed over the past two years. An invariant is any property of a geometric configuration which is unaffected by viewpoint. In current model-based vision approaches , it is necessary to test each object in the library since the specific properties of the object can only be exploited for discrimination after model pose is determined. When objects are described in terms of invariants the resulting properties can be used to index large model libraries. We have shown that invariant indexing leads to object recognition cost which grows slowly with the size of the model library.

Invariants are also an effective representation for the geometric shape of an object. We have implemented two applications of invariants, the first is using a hybrid constraint and invariant model for geographic features to locate these features in synthetic aperture radar(SAR) imagery [3]. The second application demonstrates the extension of invariant theory to multiple views and arbitrary 3D pointsets [6].

This latter application appears very promising for the construction of site models without the need for determining camera models or to determine the 3D configuration of the elements comprising the site. This work is the result of a collaboration [1] with a number of researchers interested in the theory and application of invariants to image analysis and photogrammetry.

We shown that models can be transferred to any arbitrary view once as few as eight correspondences are known. These results are based on the development of Longuet-Higgins [4] who developed a linear algorithm for structure from motion. The paper in this proceedings shows that accurate model transfer can be achieved from features derived by image segmentation.

1.1.3 Image Registration

Another algorithm arising from the basic Longuet-Higgens result is an effective non-iterative algorithm for image registration and camera calibration [2]. The ap-

*Work at GE was supported in part by the DARPA Strategic Computing Vision Program and the Air Force Office of Scientific Research under Contract No. F49620-89-C-0033 and by the DARPA Strategic Computing Vision Program under Contract No. MDA972-91-C-0053.

[1] The individuals involved in the collaboration are, E. Barrett and P. Payton of Lockheed Missles and Space Division, M. Brill of SAIC, McLean VA.

[2] See a more detailed treatment, "Calibration of Cameras Using the Essential Matrix," by R. Hartley, in this

plication considered in this work is the problem of determining camera parameters and the structure of 3D features from two or more unknown viewpoints.

The original work of Longuet-Higgens assumes that the internal camera parameters are given and the problem is to determine the external pose parameters of the cameras. Unfortunately it is often difficult to obtain accurate estimates of the internal parameters and existing camera calibration codes are often iterative and there is a non-negligible probability of the process failing to converge. Hartley is able to determine the two camera focal lengths as well as the five relative external parameters. The determination of focal length is often the most unstable process in conventional camera modeling programs.

The approach taken by Hartley is non-iterative which means that the algorithm can be used as a reliable component in recognition or site modeling systems. Recent experiments have demonstrated that for the case of stereo pairs the camera models derived by the algorithm resulted in image registration accuracy of better than .15 pixels.

1.2 Object-Oriented Design

In addition our research and applications of object recognition techniques, we are involved in a number of projects for the application of object-oriented design to the development of research and application environments.

1.2.1 CMEE

As part of the DARPA sponsored RADIUS project, it is planned to develop a common software environment(RCDE) to facilitate the exchange of results and to provide a platform for the demonstration of algorithms which are targeted at the intelligence exploitation of aerial images. Currently, the Cartographic Modeling Environment(CME) which is under development at SRI International is being extended and ported the UNIX operating system from the Symbolics Lisp Machine. GE and SRI have teamed to provide the documentation and extensions to CME to meet the requirements of the RCDE[3]

1.2.2 IUE

A major DARPA project has been initiated to provide a standard software environment for carrying out image understanding research and applications. The Image Understanding Environment or IUE is currently being specified by a committee of senior IU researcher and is based on an object-oriented representation of the major data structures and operations used in IU algorithms.

The goal is that the IUE will become a standard of exchange of IU data and also provide common interfaces so that new algorithms and other support code can be freely exchanged between research institutions. The current state of affairs is that code is often duplicated and it is very difficult to evaluate algorithms developed in one environment on another.

The IUE specification is scheduled to be complete by February 1992. An integrating contractor will be selected in 1992 and the first prototype of the IUE will be available for distribution by the end of 1994. The prototypes will be evolved by a rapid-prototyping strategy and will use code developed at IU research institutions under the specification guidelines and supervised by the integrating contractor.

The IUE project is described in more detail in this proceedings [4]

1.2.3 Geometer

Most of our applications and research systems have been built on top of a set of geometry and algebra tools called GEOMETER. We have integrated GEOMETER with CME on the Symbolics Lisp Machine to provide a fairly complete environment for experimenting with constraint-modeling and object recognition. Currently we are porting our GEOMETER systems to the C++ language and extending the functions of GEOMETER to handle computations involving invariants. We are attempting to keep the class hierarchy of GEOMETER++ consistent with CMEE so that we can reintegrate the two systems after the porting process is stable.

In the remainder of this report we provide more details about the work and some highlights of the experiments and demonstrations over the eighteen months.

2 Constraint-Modeling

Our underlying approach to constraint modeling is the observation that all geometric constraints can be represented in terms of algebraic equations. A constraint-model is thus a system of polynomial equations and specific model instances correspond to root of the polynomial system. In several earlier papers we have provided details of the representation and have shown results on a number of solution techniques [9, 8]. Two of significant advancements have been made in the solution of constraint equations since our last report in the IU workshop proceedings.

2.1 Exploiting Sparse Matrices

We have discovered that most of the equations which result from defining a set of constraints yield a sparse coupling between variables and equations. When defining geometric constraints the variables are associated with parameters of the geometric primitives such as points, surfaces and curves. The equations defined by geometric relations set constraints on the variables, but in most cases, a given variable only appears in a small fraction of the equations.

For example if two planes are required to be perpendicular, the corresponding equation is $\mathbf{n}_1 \bullet \mathbf{n}_2 = 0$ If there is an additional perpendicularity constraint with respect to a third plane, $\mathbf{n}_1 \bullet \mathbf{n}_3 = 0$, it does not involve any of the parameters associated with the second plane.

proceedings.

[3]See the review of RCDE in this proceedings, "The RADIUS Common Development Environment", by J.L. Mundy et al.

[4]"The Image Understanding Environment Project", J.L. Mundy et al.

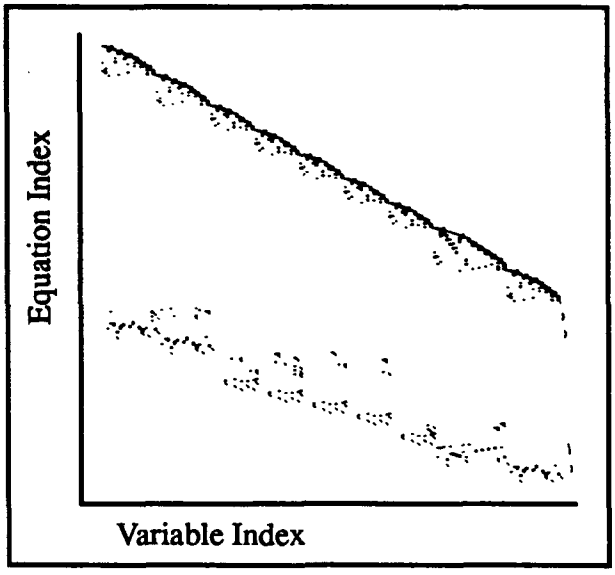


Figure 1: The entries of the Jacobian matrix of the set of constraints for the hangar building. The figure shows that constraints are sparse.

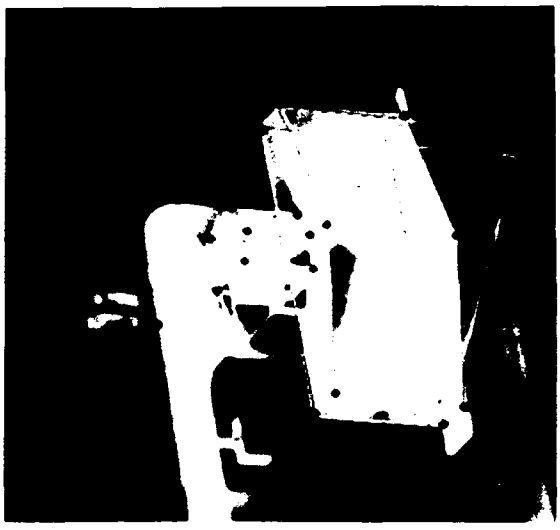


Figure 2: The use of constraints to model an aircraft hangar. The figure shows the convergence process. The set of lines extending from the building outline correspond to the model when the model vertices are set to the corresponding image locations. After convergence, the entire model is indistinguishable from the image boundaries.

Figure 1 describes the non-zero elements in the Jacobian matrix for a typical constraint modeling example. The top half is block-diagonal, has equations for enforcing the planarity of the faces for all the components in figure 2. We have been able to exploit the sparse nature of the constraint system by implementing the solver using sparse matrix techniques. When the matrices are sparse, the processing time can be reduced from $O(N^3)$ to $O(n)$. This approach has allowed us to solve large constraint systems with up to a thousand variables.

2.2 Avoiding the Constraint Surface

The classical approach to solving constrained optimization problems is to expand the constraint surface into a first or second order polynomial and then move along the constraint surface in a direction defined by the gradient of the cost function while staying on the surface. We have found that this approach leads to slow convergence and difficulties with singularities.

In the constraint-based modeling application, a much better strategy is to set as many of the model parameters as possible to values determined from image observations. The process is described in more detail as follows.

The two goals of finding the global minimum of a convex function, $\nabla f(\mathbf{x}) = 0$, and satisfying the constraints, $\mathbf{h}(\mathbf{x}) = 0$, are combined to give an over-constrained problem, whose linear approximations give:

$$\begin{aligned} \nabla^2 f(\mathbf{x})\,\mathbf{dx} &= -\nabla f(\mathbf{x}) \\ \nabla \mathbf{h}(\mathbf{x})\,\mathbf{dx} &= -\mathbf{h}(\mathbf{x}) \qquad (\sqrt{c}) \end{aligned} \tag{1}$$

Since the two goals are in general conflicting, only the least-square-error solution can be solved. The constraint equations are multiplied by a factor $\sqrt{c}$. Each iteration of (1) has a line search that minimizes the least-square-error:

$$m(\mathbf{x}) = |\nabla f(\mathbf{x})|^2 + c\,|\mathbf{h}(\mathbf{x})|^2 \tag{2}$$

Starting with $\sqrt{c} = 0$, system (1) converges to the unconstrained global minimum first, and so avoids local minima and singularities on the constraint surface. As noted before, this convergence is trivial, because the fitting error $f(\mathbf{x})$ is well approximated by a quadratic, and so the Hessian $\nabla^2 f$ is almost constant. It takes only one iteration for affine projection, and a few more for perspective and more nonlinear projections.

When the Hessian $\nabla^2 f$ is constant, the best-fit surface is a linear subspace with zero curvature and no singularity, a lot simpler than the constraint surface. The independent variables of the best-fit surface corresponds to depths of the visible vertices which can be freely varied as long as their projections agree with the image.

All the vertices can be fully constrained if the model is fit to more than one view, in which case best-fit fully solves the shape similar to photogrammetric stereo. Even with one view, most of the shape is specified, and this helps avoid local singularities such as collapsing locations and orientations.

One can slide on the best-fit surface looking for the point nearest to the constraint surface. This can be done by first eliminating variables using the gradient equations, then using the remaining variables to solve for least residual $|\mathbf{h}(\mathbf{x})|$ of the constraint equations. Alternatively, system (1) is solved with $\sqrt{c} \ll 1$ with elimination done largest pivot first.

The convex fitting function $f(\mathbf{x})$ can be viewed as a regularizer in the solution of $\mathbf{dx}$. It makes the constraint problem well-posed by using empirical data whenever additional constraints are needed to pin down free variables in $\mathbf{h}(\mathbf{x}) = 0$.

3 Geometric Invariants

An invariant is a property of a set of geometric forms which does not change with viewpoint. Our premise is that invariants offer a sound framework for the representation of objects leading to efficient recognition algorithms. For the past few years we have worked jointly with Oxford University to develop and apply geometric invariants to the problem of object recognition [5]. A joint workshop between DARPA and ESPRIT, "Applications in Computer Vision," was held in Reykjavik, Iceland in April, 1991. The workshop brought together the leading researchers in invariant theory and applications. A collection of the papers from the workshop is currently in preparation and will be published by MIT Press [7]. Below are some of the highlights our recent results in the application of invariants.

3.1 Invariant Indexing Functions

We have recently finished a first prototype recognition system for planar objects which can retrieve object classes in nearly constant time even under extreme perspective effects and scene complexity. [10]. The majority of the indexing functions used to date in model based vision are planar projective invariants. In our current implementation (an extension of [11]), three algebraic invariants are used.

Invariant 1: Five Coplanar Lines.

Given five coplanar homogeneous lines $\mathbf{l_i}$, $i \in \{1,..,5\}$, two projective invariants are defined:

$$I_1 = \frac{|M_{431}||M_{521}|}{|M_{421}||M_{531}|} \qquad I_2 = \frac{|M_{421}||M_{532}|}{|M_{432}||M_{521}|} \qquad (3)$$

where $M_{ijk} = (\mathbf{l_i}, \mathbf{l_j}, \mathbf{l_k})$ and $|M|$ is the determinant of M. Should any triple of lines become concurrent the first invariant is undefined. This singular case is common for polygons where alternate sides are parallel. In these cases we can only use the second invariant as a shape descriptor.

Using the duality of points and lines the invariants can be defined for five coplanar points. An alternative definition appears in [5].

Invariant 2: A Conic and Two Lines.

A conic, C, and two lines not tangent to the conic define a single invariant given by [2]:

$$I = \frac{(\mathbf{l_1^T} C^{-1} \mathbf{l_2})^2}{(\mathbf{l_1^T} C^{-1} \mathbf{l_1})(\mathbf{l_2^T} C^{-1} \mathbf{l_2})} \qquad (4)$$

where C is the matrix of the conic: a conic takes the form $ax^2 + bxy + cy^2 + dx + ey + f = 0$ which can be expressed as the quadratic form:

$$[x, y, 1] \begin{bmatrix} a & \frac{b}{2} & \frac{d}{2} \\ \frac{b}{2} & c & \frac{e}{2} \\ \frac{d}{2} & \frac{e}{2} & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0, \qquad (5)$$

or as $\mathbf{x^T} C \mathbf{x} = 0$. By duality a conic and two points also yield a single invariant.

[5]The individuals from Oxford University involved in the collaboration are A. Zisserman, D. Forsyth (now at the University of Iowa) and C. Rothwell

Invariant 3: A Pair of Conics.

Using the notation of a conic matrix C_i in example 3 above, two projectively invariant measures can be formed for a pair of conics normalized so that $|C_i| = 1$. These are:

$$I_1 = Trace[C_1^{-1} C_2] \quad \text{and} \quad I_2 = Trace[C_2^{-1} C_1] \qquad (6)$$

3.2 The Recognition System

The major functions of the object modeling and recognition process are summarized as follows.

1. **Feature extraction.** The conics and lines needed to form the invariants we use are extracted from image edge data.
2. **Model Construction** A set of features is associated with a particular object by providing one or more images of the object by itself. The invariant feature vector is computed for the set of features and the model is added to the library according to the feature vector index. The model features are also stored in the library to be used during hypothesis verification.
3. **Hypothesis generation.** The invariants for groups of features are computed. We index the measured invariants against invariant values in the library using a hash table, and if they match, produce a recognition hypothesis. Hypotheses for common objects are then combined before verification.
4. **Hypothesis verification.** When a potential match is found we confirm it by projecting edge data from an acquisition image to the test scene. Should the projected and scene edge data be sufficiently close the match is confirmed. The parameters of this projection can be computed from the geometry of the same features used to generate the invariant index. It is always the case that enough feature constraints are available to compute the transformation when the features define an invariant. The invariant vector index also uniquely defines the correspondence between model features and image features needed to solve for the transformation.

An example of the system performance is shown in figure 3. Here a bracket is found in two orientations under occlusion and severe perspective distortion. The system currently has 31 models in the library and recognition time increases slowly with the size of the library due to more false hypotheses, but the growth rate is far slower than the typical linear increase expected when each model has to be tested individually.

3.3 Canonical Frames

As just illustrated, there is considerable benefit in using invariants to imaging transformations as *indexing functions* for generating recognition hypotheses. Indices should be local and have some redundancy (i.e. several per outline) - so if one index is occluded there is a good chance recognition can proceed on other visible

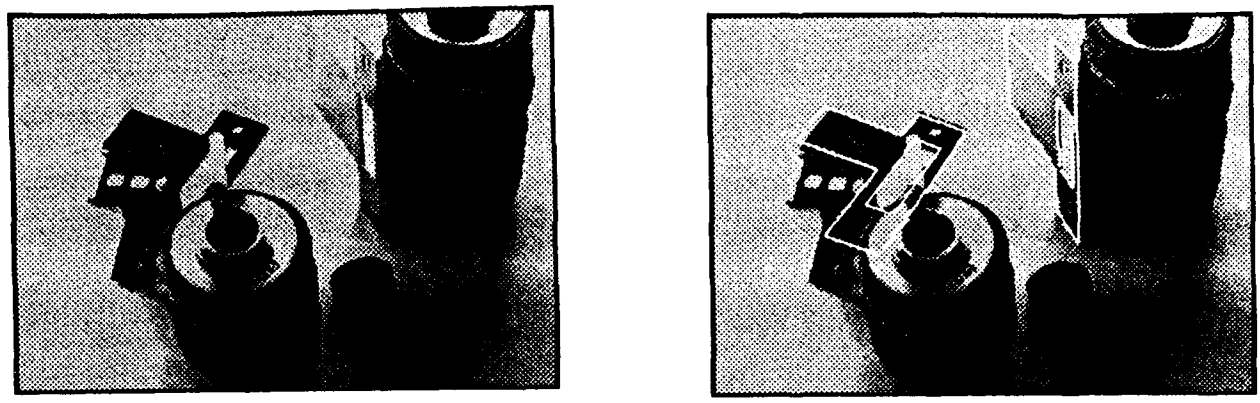


Figure 3: Recognition using projectively invariant indexing functions.

parts; they should be stable, so small perturbations in the curve (due to image noise) do not cause large fluctuations in index value; and they should have sufficient discriminatory power over models in the library (so all models do not have similar index values).

All of these requirements are satisfied by a bitangent construction, where a bitangent is a line tangent to a curve at two points on the curve. Provided the object outline is sufficiently rich in structure there will be several such constructions for each object - and thus redundancy in the representation giving partial immunity to occlusion.

We use tangency to select 4 distinguished points on the curve as described in figure 4

Figure 5 demonstrates this process for one concavity of a spanner. The point is that the object curve, and any projective view of it, are mapped into the same curve. Consequently, any (metric) measurements made in this frame are invariant descriptors and hence may be used as index functions to recognize the object. figures 5.

3.4 Rotationally Symmetric Surfaces

3.5 Recognising rotationally symmetric surfaces from their outlines

This section sketches techniques used to recognise rotationally symmetric surfaces from an outline. The work is described in more detail in an internal report [1]. The outline of a surface in an image is given by a system of rays through the camera focal point that are tangent to the surface. The points of tangency of these rays with the surface form a space curve, called the *contour generator.*

Points on the contour generator are distinguished, because the plane tangent to the surface at such points passes through the focal point (this is an alternative definition of the contour generator). As a result, we have:

> **Lemma:** Except where the image outline cusps[6], a plane tangent to the surface at a point on the contour generator (by definition, such a plane passes through the focal point), projects to a line tangent to the surface outline, and conversely, a line tangent to the outline is the image of a plane tangent to the surface at the corresponding point on the contour generator.

As a corollary, we have:

> **Corollary 1:** A line tangent to the outline at two distinct points is the image of a plane through the focal point and tangent to the surface at two distinct points, both on the contour generator.

This yields useful relationships between outline properties and surface properties. For example:

> **Corollary 2:** The intersection of two lines, bitangent to the outline is a point, which is the image of the intersection of the two bitangent planes represented by the lines.

The lemma and both corollaries follow immediately from figure 6.

Generic surfaces admit one-parameter systems of bitangent planes, so we can expect to observe and exploit intersections between these planes. One case in which the bitangent intersections are directly informative occurs when the surface is rotationally symmetric. In this case, the envelope of the bitangent planes must also be rotationally symmetric, and be ruled. The rotational symmetry follows from the rotational symmetry of the surface, and the fact that the envelope is ruled follows by considering a plane section of the surface through its axis of symmetry. Notice that, as figure 6 shows, the image outline generically has *no* symmetry in this case.

Because it is rotationally symmetric and ruled, the envelope is either a right circular cone, or a cylinder with circular cross-section (this is a right circular cone whose vertex happens to be at infinity). We shall draw no distinction between vertices at infinity and more accessible vertices, and refer to these envelopes as *bitangent cones.* These comments lead to the following

> **Key result:** The vertices of these bitangent cones must lie on the axis (by symmetry), and

[6] We ignore cusps in the image outline in what follows.

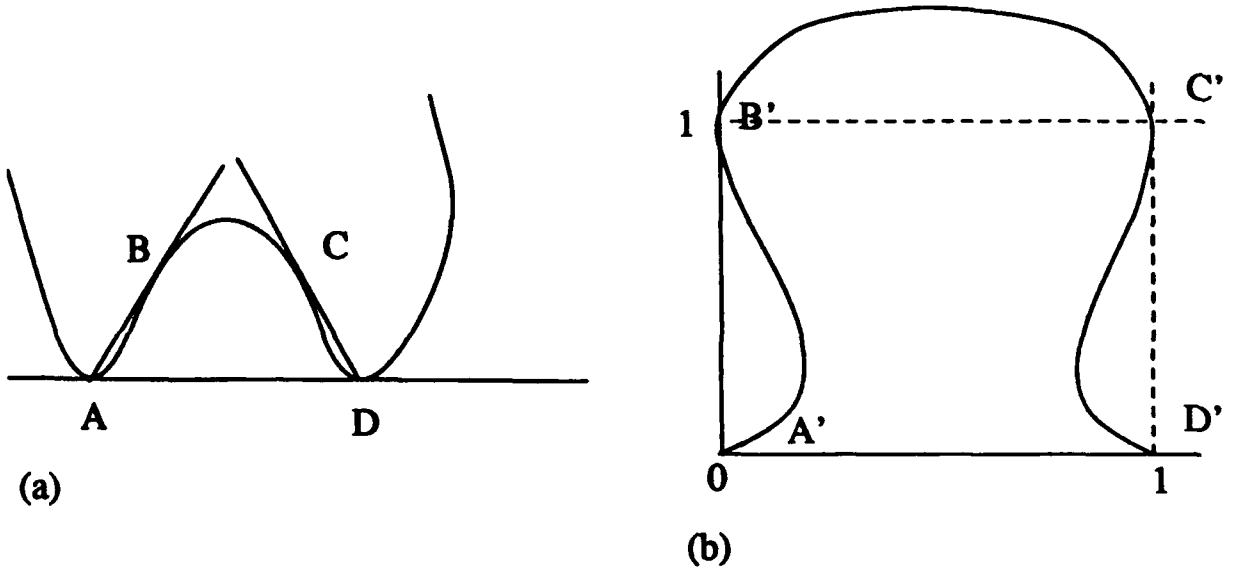


Figure 4: a. Construction of the four points necessary to define the canonical frame. The first two points (A D) are the points of bitangency that mark the entrance to the concavity. Two further distinguished points, (B C), are obtained from rays cast from the bitangent contact points and tangent to the curve segment within the concavity. These four points are used to map the curve to the canonical frame. b. Curve in canonical frame. A projection is constructed that transforms the four points in a) to the corner of the unit square. The same projection transforms the curve into this frame.

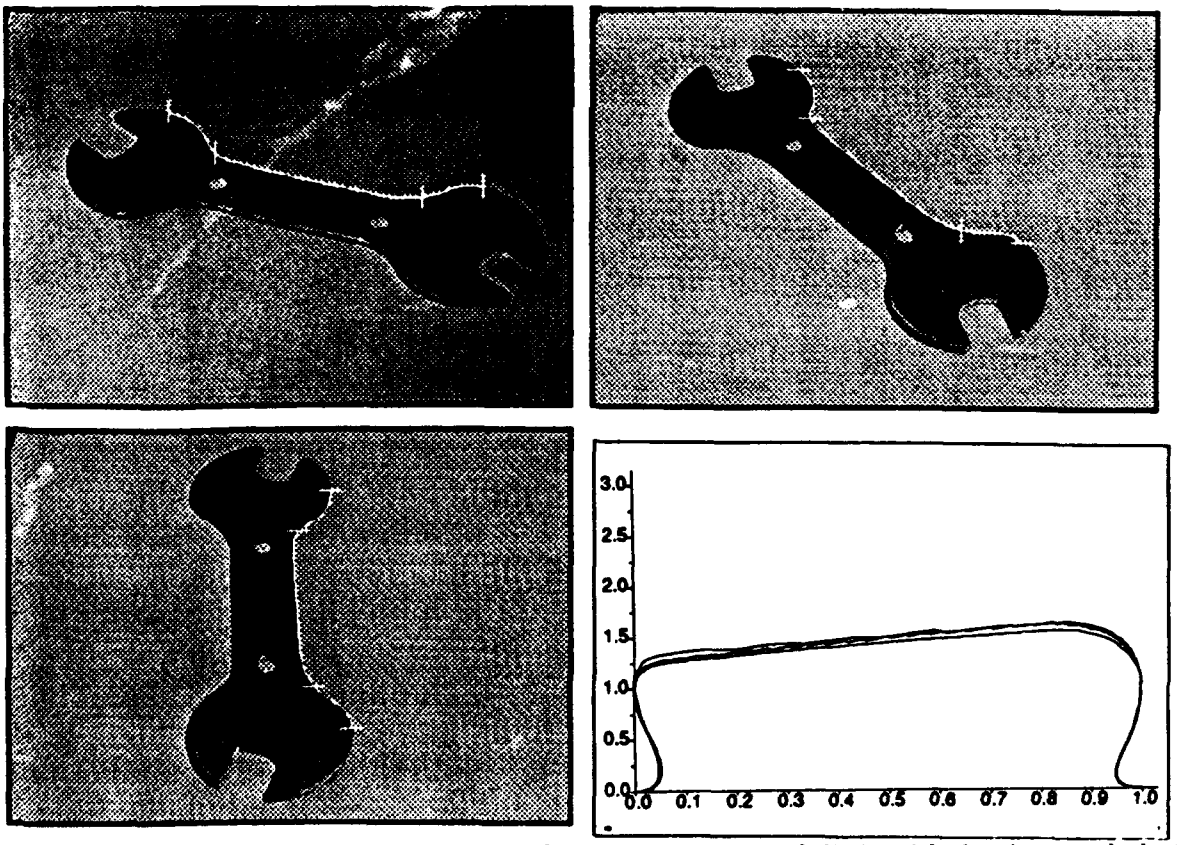


Figure 5: a) - c) 3 views of a spanner with extracted concavity curves and distinguished points marked. Note the very different appearance due to perspective effects. d) Canonical frame curves for the three different views of a spanner. This demonstrates the stability of the method. Of course the same curve would result from a projective transformation between the *object* and canonical frame.

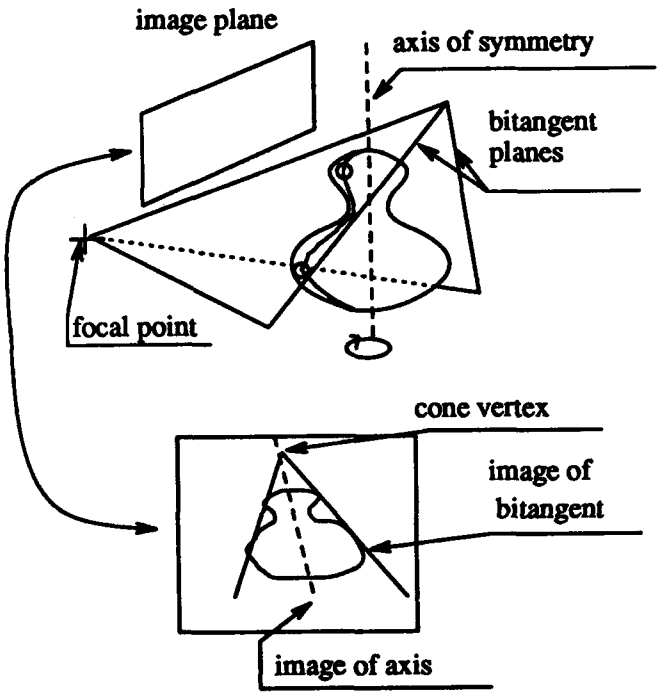


Figure 6: A rotationally symmetric object, and the planes bitangent to the object and passing through the focal point, are shown. It is clear from the figure that the intersection of these planes is a line, also passing through the focal point. Each plane appears as a line in the image: the intersection of the planes appears as a point, which is the image of the vertex of the bitangent cone. Note in particular that the image outline *has no symmetry*. This is the generic case.

so are collinear. Assuming the focal point lies outside the surface, as figure 6 shows, the vertices of the bitangent cones *can be observed in an image*. The vertices appear as the intersection of a pair of lines bitangent to the outline.

As a result, if the surface has four or more bitangent cones, the vertices yield a system of four or more collinear points, lying on the axis of the surface. These points project to points *that can be measured in the image*. This fact yields two important applications:

- Cross-ratios of the image points, which are projectively invariant, yield indexing functions for the surface, which can be determined from the outline alone. These indexing functions can be used to recognise the surface.
- The image points can be used to construct the image of the axis of a rotationally symmetric surface from its outline.

Since a change in camera parameters simply changes the details of the projection of the points, but does not change the fact that the map is a projection, these cross-ratios are invariant to changes in the camera parameters, as well as to camera position and orientation.

This approach can be generalized in two ways. Firstly, there are other sources of vertices than bitangent lines. Secondly, the geometrical construction described works for a wider range of surfaces than the rotationally symmetric surfaces.

At present, there are four further known sources of vertices, illustrated in figure 7:

- **The tangents at a crease in the outline:**
- **The tangents at an ending in the outline:**
- **A tangent that passes through an ending in the outline:**
- **Inflections of the outline:**

In each case, there is a clear relationship between the tangent to the outline and a plane tangent to the surface. In each case, the envelope of the system of planes tangent to the surface and having the required property, is a cone with a vertex along the axis. These sources of information are demonstrated in figure 7.

All the constructions described for vertices are preserved under projectivities of three-space: this means that, for example, given a plane tangent to a surface at two points, a projective mapping takes the surface to a new surface, and the bitangent plane to a plane bitangent to the new surface, where the points of tangency are given by applying the map to the original points of tangency. As a result, all these constructions apply to surfaces that are projectively equivalent to rotationally symmetric surfaces, for example, surfaces obtained by sweeping a given ellipse along an axis, and scaling it while sweeping it. Figure 8 shows two views of the base of a lamp, with bitangents marked. These figures yield values of the cross-ratio within under 5% of one another. Thus, a cross-ratio formed in this way is a useful indexing function.

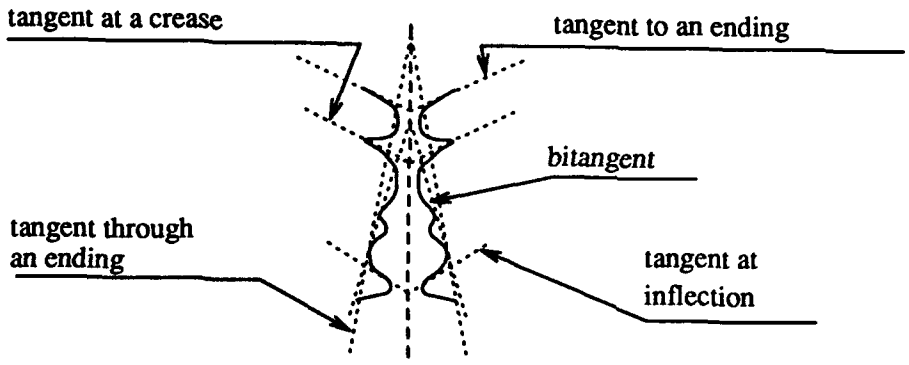


Figure 7: The five known cases that produce usable coaxial vertices. Note that although this figure appears to have a reflectional symmetry, this is not a generic property of the outline of a rotationally symmetric object.

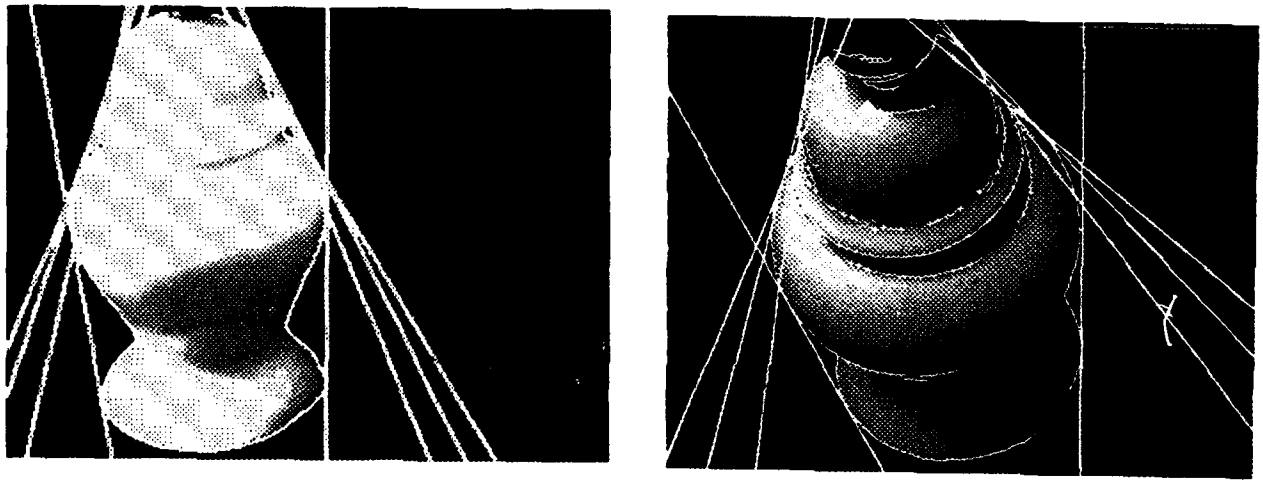


Figure 8: Two views of a lamp base, illustrating bitangent construction

References

[1] D. A. Forsyth, J. L. Mundy, A. Zisserman, and C. A. Rothwell. Recognising curved surfaces from their outlines. Report, University of Oxford, October 1991.

[2] David Forsyth, Joseph L. Mundy, Andrew Zisserman, Chris Coelho, Aaron Heller, and Charles Rothwell. Invariant descriptors for 3d object recognition and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1991. To appear.

[3] Aaron J. Heller, David M. LaRocque, and Joseph L. Mundy. The interpretation of synthetic aperture radar images using projective invariants and deformable templates. In *Proc. DARPA Image Understanding Workshop*, 1990.

[4] H. C. Longuet-Higgens. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–235, 1981.

[5] R. Mohr and L. Morin. Relative positioning from geometric invariants. In *Proceedings CVPR*, pages 139–144, 1991.

[6] J. L. Mundy, M. H. Brill, P. M. Payton, E. B. Barrett, and R. P. Welty. 3-d model alignment without computing pose. In *Proc. DARPA Image Understanding Workshop*, 1992.

[7] J. L. Mundy and A. Zisserman, editors. *Geometric Invariants in Computer Vision*. MIT Press, 1992. To Appear.

[8] V. Nguyen, J.L. Mundy, and D. Kapur. Modeling Generic Polyhedral Objects with Constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 479–485, Lahaina, Hawaii, June 1991.

[9] V.-D. Nguyen, J. L. Mundy, and D. Kapur. Modeling polyhedra by constraints. In *Proc. DARPA Image Understanding Workshop*, 1990.

[10] C. A. Rothwell, A. Zisserman, D. A. Forsyth, and J. L. Mundy. Efficient model library access by projectively invariant indexing functions. Report, University of Oxford, 1991.

[11] C. A. Rothwell, A. Zisserman, D. A. Forsyth, and J. L. Mundy. Using projective invariants for constant time library indexing in model based vision. In *BMVC*, 1991.

IU AT UI: AN OVERVIEW OF RESEARCH DURING 1990-91

Narendra Ahuja and Thomas Huang
Beckman Institute and Coordinated Science Laboratory
University of Illinois
405 North Mathews Avenue
Urbana, Illinois 61801

Abstract

This paper presents an overview of the research in image understanding (IU) at the University of Illinois (UI) conducted during 1990-91. During this period, we have made progress in four areas: integration in three-dimensional vision, motion analysis, recovery based scene visualization, and representation and navigation. Work in each of these areas is reviewed.

1. Introduction

A major part of our recent research is in four areas of image understanding. The first area (Sec. 2) deals with integration of multiple image cues in performing image interpretation. These cues capture different aspects of the scene structure, and their integrated analysis leads to a more robust inference about the scene characteristics than possible from individual cues. The second area (Sec. 3) is concerned with our work on interpretation of image sequences showing dynamic scenes. Here we consider the problems of detecting feature correspondences and estimating the three-dimensional (3D) motion parameters and the 3D surface structure from feature correspondences, over a sequence of images showing rigid or nonrigid motion. Projects in the third area (Sec. 4) report work on visualization of scenes using attributes recovered during interpretation, or depiction of 3D characteristics using artificial attributes. The use of image attributes during 3D recovery for visualization takes advantage of the most powerful image cues of scene structure for perceptually effective scene representation. The fourth area (Sec. 5) is concerned with different components of an evolving 3D representation and navigation system which has the goal of autonomously acquiring, maintaining and using 3D information about the environment. Representative projects in each of these areas are summarized in the following sections. To keep the paper brief, we have minimized discussion of and references to relevant work done by others. Such discussion and references are available in the cited and other listed publications.

*Parts of this research were supported by the Defense Advanced Research Projects Agency and the National Science Foundation grants IRI-8902728, Air Force Office of Scientific Research under grant AFOSR-90-0061, Army Research Office Advanced Construction Technology Center under grant DAAL 03-87-K-0006, Joint Services Electronics Program under grant N00014-90-J-1270, and the State of Illinois Department of Commerce and Community Affairs under grant 90-103.

2. Integration

Our goal in this area is to perform 3D or other interpretation of images, such that the interpretation simultaneously satisfies a range of constraints imposed by the image structure and the model of the scene. To do this, we use different computational processes each of which carries complementary or redundant information derived from different image cues. Image interpretation is the result of a cooperative computation that resolves conflicts and ambiguities arising from the individual processes. We have presented several examples of the integration approach in previous IU workshops [36, 37]. One formalism for unifying our work on integration is presented in [35, 20]. Here we summarize some recent work on integration.

2.1. Integrated Active Stereo

The goal of active stereo is surface reconstruction from stereo images of large scenes having large depth ranges, where it is necessary to aim cameras in different directions to fixate at different objects and to construct the global surface map of the scene from small patches. The first stage of this work involved surface reconstruction of a single object, having no depth discontinuities. It performs integration of camera vergence, focus, aperture, stereo and calibration processes [37, 4]. Early work on the second stage, involving arbitrarily placed and arbitrary size objects was reported in [19, 18]. This second stage has now been completed [22, 24]. In this stage, a part of the visual field that has not yet been fixated but has appeared as the peripheral visual field during a fixation, provides coarse (inaccurate) structural information, to be refined during future fixations. The availability of coarse peripheral maps make it possible to se-

lect a new fixation point on a new object. The process of fixation has three components: moving the optic axes so they intersect at the fixation point (vergence control), registering the sharpest image of the fixation point (focus control), and obtaining the best depth estimate of the fixation point by optimal combination of the estimates from focus, vergence, and stereo. Coarse-to-fine images are acquired and analyzed during the fixation process thus representing integration of coarse-to-fine image acquisition and coarse-to-fine surface reconstruction. The coarseness of the images results naturally from optical blurring in the vicinity of the target point during the current fixation. The depth estimates obtained during fixation from focus and vergence are used as independent observations in addition to stereo to reduce the uncertainty in the depth of the fixation point. Analytical evaluation of the performances of stereo, vergence, and focus is used to devise optimal strategies for their integration. The final depth estimate from stereo, focus and vergence is obtained in a way such that the uncertainty of the fused estimate is less than that of the individual depth estimates. The various objects in a scene are represented by the vertices of a graph. The problem of surface reconstruction for the scene then amounts to fixating these objects successively and scanning their surfaces. Assuming that no object is fixated twice, the order of fixation is determined by finding a path in the graph. If a cost is assigned to every change of fixation from one object to another, then the solution is one of finding the minimal cost path in the graph. However, since the number of objects and their spatial relationships are unknown until the scene has been completely scanned, only a locally minimal cost path is determined.

2.2. Integrated 3D Mapping and Calibration

The work just described involves integration of the camera calibration process with surface reconstruction. Thus, the camera calibration parameters are treated as variables (rather than constants given by the encoder readings), which makes camera calibration process an integral, flexible component of surface reconstruction. As a result, despite using ordinary cameras prone to calibration errors during movements, the integration could achieve an average absolute estimated depth error of less than 0.15% for a large surface having a depth of approximately 2 meters [4].

The above loss of calibration does not involve monotonic accumulation of errors because the camera movements are periodic with bounded errors. However, during unrestricted movement of a mobile platform housing the cameras, the calibration errors may accumulate due to factors such as wheel slippage. Such errors can only be corrected using external spatial references. Two such references are: partial reconstructions of scene surfaces and recognizable landmarks. To use these requires that the scene information acquired from different camera configurations and viewpoints be combined for integrated estimation of scene structure and camera motion.

We have now addressed this general problem of integrating piecewise scene descriptions derived from different viewing configurations, under uncertainities of the knowledge of viewing parameters [30]. The type of descriptions, the viewing parameters undergoing change, and the nature of error sources may vary but many elements of our treatment of the problem would apply. Specifically, we have considered the problem of a roving active stereo system which acquires partial depth maps of the scene from different viewpoints. The surface information obtained from dynamic cameras involves error accumulation during extended motion. Generally, the amount of deviation is proportional to the distance travelled. Therefore, while the information from a single viewpoint can provide reliable localized surface map, the accuracy of the global map decreases continuously. We have studied the effect of using a large number of the most recently acquired images and the corresponding incremental scene information to update the global map, instead of using only the most recent single frame. This work has obvious overlap with the work on navigation reviewed in Sec. 5.

2.3. Integrating Region, Border and Component Gestalt for Extracting Perceptual Structure

This research concerns perceptual grouping, or goal independent detection of perceptual organization in images. Elsewhere, we have reviewed our approach to perceptual grouping of dots that integrates multiple constraints, active at different perceptual levels and having different scopes in the dot pattern [37]. We have investigated two extensions to this work [33]. The first is the use of human training to identify the various perceptual roles that a dot can play. The second is the use of systematic feature selection techniques to identify the most relevant features in the computation of perceptual grouping. Here, we have concentrated on the identification of interior and border dots. A total of 23 features are initially used to characterize the geometric properties of the Voronoi polygons defined by the dots. These features include the moments of area and the various invariants computed from them, compactness, elongation, and eccentricity. By performing feature selection, we have shown that the following six features are the most useful: (a) eccentricity magnitude, (b) x direction and (c) y direction of the eccentricity vector, (d) the elongation magnitude, (e) the x direction and (f) the y direction of the major axis of the cell. The assignment of perceptual roles is accomplished by an initial classification using the training data and the nearest neighbor method in the feature space. This initial non-contextual classification is then refined using a probabilistic relaxation technique that uses Gestalt criteria such as border smoothness.

3. Motion Analysis

The long-range goal of our research in this area is the understanding of dynamic scenes. We have made progress in three major subareas: finding feature correspondences in image sequence, determining rigid motion parameters and surface structure from the correspondences, and analyzing nonrigid motion.

3.1. Detecting Feature Correspondences

Detecting feature correspondences is difficult due to a wide variety of three-dimensional structural discontinuities and occlusions that occur in real world scenes. Our work on this problem is concerned with matching point features and is divided into two categories, according to whether the correspondences are detected in two, long range views, or in a dense image sequence.

In the first category, we have developed a matching algorithm for obtaining feature point correspondences across images containing rigid objects undergoing different motions [31]. The algorithm is applied recursively to coarse-to-fine resolution images. At each resolution, a point feature detector derived from the Moravec interest operator is applied to obtain a number of distinguished feature points. Then a variety of matching constraints are applied sequentially, starting with simplest and following with more informed ones. First, an intensity-based matching algorithm is used to obtain unique point correspondences. This is followed by the application of a sequence of newly developed constraints involving rigidity and disparity. The rigidity tests match geometrical relationships among the feature points, while the disparity test ensures that no matched feature point in an image could be rematched with a different feature if it were reassigned a disparity value associated with another matched pair. The computational complexity at each resolution level is proportional to the number of detected feature points in the two images at that level. Experimental results with several kinds of indoor and outdoor scenes show that the algorithm yields only correct matches for scenes containing rigid objects.

We have initiated a research project in applying concept learning to matching [38]. Much significant work has been done in the field of machine learning in terms of producing excellent classifiers. All of this work can be brought to bear on the matching problem. By exploiting the information in the template image, we can derive a set of negative examples. Specifically, if there is any ambiguity in matching to the object image, it will be caused by similar points in the object image in terms of the error measure used for matching. Since there is overlap between the template and object images, the points which would have caused problems in direct matching should exist in the template image. Thus, we find points similar to the template point. We know that all of these similar points are sources of ambiguity so we label them as negative examples, and the template point is the positive example. Then we can pass this set of examples to a machine learning algorithm which will learn a concept of the template point. Finally, we classify each point in the search area in the object image, and select the point which classifies closest to the template point as the correct match. The preliminary empirical results on a stereo pair of a street were promising at 96% correct matches of zero-crossings. Testing the training data matrix with a neural network concept learner, a specializer, and a generalizer are scheduled next.

In the second category, we have developed a sequential hypothesis testing approach to find trajectories of feature points in a dense image sequence. This is based on our earlier work on detection of small, faint moving objects from many frames. More details can be found in a separate paper in the proceedings [32] and in [34].

3.2. Rigid Motion and Structure from Two-View Correspondences

Our work in this area is concerned with estimating motion and structure of a scene from feature correspondences over two views showing long term motion.

We have developed an integrated system for 3D motion analysis and object recognition with outdoor stereo images as inputs [41]. The goals are to obtain the 3D motion description and the identification of the object in the input stereo images. We use stereo images for motion estimation and only the left image of a stereo image pair for the processes related to object recognition. The system consists of four stages. These are (i) motion estimation, (ii) distinctive feature extraction, (iii) model database, and (iv) object recognition. The motion estimation is based on 3D point correspondences which can be derived from matched points on the stereo images. In this stage, we obtain the following: (1) motion parameters, (2) 3D centroid locations of the object and (3) the region of interest (region of the object projected on an image). Both (2) and (3) are used in other stages to extract properties of the object on the input images. For object recognition, we use two features that can be extracted consistently from a grey-level image to describe an object. These two features are the region of interest and the wheel-pattern of a vehicle. The first feature, region of interest, can be obtained from motion detection while the wheel-pattern is extracted by a modified Hough transform algorithm. The model database is used to generate the perspective view of each model library for comparison with the input image. It requires four parameters (roll, azimuth, elevation and distance) to generate a perspective view. These four parameters can be derived from the 3D centroid locations and translation directions obtained in motion estimation. ¿From the generated views, we know the wheel-pattern and the region of interest of each model library. In addition, the expected size of wheel can be found and used in the wheel detection algorithm to reduce the computation time. With the required features, namely wheel-pattern and region of interest, of the input image and each model library, we can generate the attribute list sets of the image and the model libraries. A typical at-

tribute list set has four elements which are (i) region of interest, (ii) number of wheels, (iii) locations of wheels and (iv) size of wheels. By comparing the attribute list set of the image with the attribute list set of each model library, a confidence measure can be computed. The vehicle in the input images is identified to be one of the model libraries represented by its corresponding perspective view with the highest value in confidence measure. We have used this approach on sets of stereo image pairs. The experimental results show that the system can provide the 3D motion description and the identification of the vehicle in all the stereo image pairs successfully.

We have developed an algorithm which matches image regions to non-iteratively estimate 3D motion and structure of a moving piecewise planar textured surface from two perspective views [29]. The algorithm has two major steps. In the first (coarse) step, the local planar nature of the surface is used to obtain polynomial expressions for image plane displacements of features. With regions as moving features, the image is segmented using Hough transform such that the regions in each segment have the same polynomial coefficients. The values of these coefficients and region properties (e.g., area) are then used to identify region correspondences. In the second step, the region correspondences are used to compute the motion parameters and surface orientation in closed form for each planar patch. The second step uses a finer model of motion than the first step. Further, we have identified sufficient conditions for double or unique solution of the problem of motion and structure from two monocular images. We list several properties of the essential matrix and the plane motion matrix, both of which are frequently used in the motion and structure estimation problem. We have developed a robust algorithm and some essential results for plane motion estimation from two-view point correspondences. We present necessary and sufficient conditions for plane motion solution. We prove that three-view matching does not necessarily make the solution unique. Further, we have developed a two-view nonlinear motion algorithm which has the following salient features: (i) it applies to any surface undergoing rigid motion; (ii) it globally minimizes an objective function (iii) it is fast; and (iv) it can determine whether the motion is unique. This work is presented in [27, 21].

3.3. Rigid Motion and Structure from Long Image Sequences

In this area, we are concerned with long sequences of images of a dynamic scene. We are interested in segmentation of the sequence into distinctly moving objects, as well as in the estimation of their motion and structure. We will discuss two separate cases, according to whether the successive images in the sequence show long range motion or short range motion, i.e., whether the image sequence is sparse or dense.

In the first category, we have developed a new model for vehicle-type motion which assumes that the motion is a rotation around an axis through the vehicle center followed by a forward translation along the main axis of the vehicle. It helps greatly to have a motion model which includes only a small number of parameters which can be assumed constant over many image frames. For the chosen model, we have established the following results: (1) When the rotation and the amplitude of translation are constant, this type of motion is equivalent to a constant camera-centered motion. This indicates that a constant motion in the conventional camera-centered model, which is commonly considered artificial, can in fact be a reasonable model in real life. (2) A constant vehicle-type motion can be interpreted as a constant screw motion. A linear algorithm for estimating constant vehicle-type motion has been developed and applied to a sequence of images of an outdoor scene containing a moving truck. The results are far superior to those from two-view methods [47]. This work is discussed in a separate paper in these proceedings [66]
.

Another algorithm that we have developed uses the model that the rotational velocity is constant and the rotation center arbitrary. It estimates 10 parameters for motion and structure of a rigid planar patch given point correspondences in a monocular image sequence under perspective projection. The algorithm mainly consists of two steps. First, the 3D space of $(\omega_x, \omega_y, \omega_z)$ is searched exhaustively but coarsely to minimize an objective function. For each selected $(\omega_x, \omega_y, \omega_z)$, we linearly compute all the other parameters. Some of the $(\omega_x, \omega_y, \omega_z)$ and the corresponding structure values are used as the initial guesses in the second step to carry out a fine search to iteratively minimize the objective function with respect to the five variables for rotation and structure. The solution corresponding to the global minimum is used to obtain least squares estimates of the remaining unknowns - translation and rotation center. We have experimentally found that the objective function converges well so that we do not have to search the 3D space densely in the first step. Results are presented for three image sequences, two simulated and one real [28].

In the second category, we have developed an algorithm which, given a dense temporal sequence of intensity images of multiple moving objects, will separate the images into regions showing distinct objects, and for those objects which are rotating, will calculate the three-dimensional structure and motion [13] using factorization into motion and structure. The algorithm consists of two major steps: finding and tracking feature points on the objects in the images, and determining the subsets of trajectories, motion, and structure corresponding to the different objects. The feature tracking algorithm can track features across frames in which the feature is occluded or the feature detector response is weak. The trajectories are partitioned into groups corresponding to the different objects by fitting

the trajectories from each group to a hierarchy of increasingly complex motion models in a coarse-to-fine manner. Further details of this work are presented in a separate paper in these proceedings [32] and in [34].

3.4. Nonrigid Motion

We have worked on several problems that involve interpretation of image sequences containing nonrigidly moving objects such as deformable objects, articulated objects and fluids.

We are investigating the use of computer vision techniques in the nondestructive evaluation of structural damage. The structure (a bridge, for example) under study is loaded statically or dynamically, and the displacements and movements of its features are observed (by several CCD and TV cameras). These measured data are combined with a priori knowledge of structural geometry to estimate structural parameters such as stiffness. Reliable feature matching and measurements to subpixel accuracy are critical to the success of our method. We are currently experimenting with a small two-story frame model to study the feasibility of our approach [39].

In many scenarios, robots and human bodies can be modeled as articulated objects. We have developed three algorithms under the assumption of perspective projection for the recovery of 3D structure from the motion of several types of joints of articulated objects, viz. joints which allow only fixed-axis rotation and joints which allow only planar rotation [68] . Then we have applied these algorithms to the analysis of human ambulatory motion in a combined way. Our numerical experimental results with synthesized data have indicated that the real solution from the proposed algorithm was always unique up to a reflectance and the time of convergence was reasonable. Details of these algorithms are presented in a separate paper in these proceedings [67].

We are investigating the problem of interpolating, under physical constraints, 3D vector fields from sample vectors at irregular positions. This problem arises from analysis of fluid motion, but our results can also be used in such areas as geometric modeling, approximation theory, and other types of nonrigid body motion. Our algorithm combines the generalized multivariate quadratic interpolation and physical constraints into one step to form an over-determined linear equation system [55]. The least squares solution of this system gives the coefficients of interpolation. Since the interpolation is done in one step, it is non-iterative and efficient in computation. We utilize methods in robust statistics to detect outliers in the sample data so that the results are more stable in the presence of gross errors. Another merit of our scheme is that by incorporating physical constraints into linear equation system, the algorithm takes the characteristics of vector field into account and is much less sensitive to noise. The algorithm is applied to both synthesized and empirically measured 3D vector fields. With the application to 3D fluid flow in mind, we study the applicability of physical constraints according to the kinematics of fluid and analyze the sources of noise from the real data acquisition setup. A comparison of our algorithm with previous work under different noise levels shows the robustness of our algorithm.

4. Recovery Based 3D-Visualization

We have started work on using the information extracted during 3D interpretation to perform image synthesis for the visualization of the original scene. One such effort is concerned with identification and depiction of image attributes such that the display communicates effectively the 3D scene structure as seen by an observer in relative motion to the scene.

4.1. Flight Images Sequences

We have addressed the problem of visualizing motion of an observer relative to a scene. Our work in this area has two objectives. First, it addresses the problem of recovering motion and structure parameters from a monocular image sequence. Second, it uses the intermediate outputs of the recovery procedure to sythesize an image sequence that depicts the estimated motion and structure. A key feature of the approach presented is an integrated use of multiple image attributes which are shared by both estimation and visualization processes. We focus on flight image sequences, i.e., image sequences acquired by an observer moving smoothly over a planar, textured surface. The approach presented allows the use of image cues such as regions, point features, optical flow, texture gradient and vanishing line. The integration of information in these diverse cues is carried out using optimization. Visualization is done using the image attributes extracted from the image sequence as well as artificial attributes. Experimental have been conducted with a real image sequence digitized from a commercially available video tape. We have produced a video tape showing the visualization sequence. This work is described in a separate paper in these proceedings [14] and in [15] .

4.2. Heart Images Sequence

In another project on 3D estimation and visualization, we are investigating techniques of estimating the motion and deformations of left ventricle from 3D data based on the hierarchical decomposition of the motion into two parts: global movement and local movement [59]. The global movement is further decomposed into global rigid motion and global nonrigid motion while the local movement is decomposed into local rigid motion and local deformation. Such a decomposition enables us to devise a coarse-to-fine modeling of the left ventricle dynamics according to *a priori* knowledge of the heart motion patterns. We then formulate the motion and deformation analysis of the left ventricle as a series of estimation processes which correspond to these decompositions. Our algorithms are based on the 3D data points derived from biplane angiogram sequences. We

first estimate the global motion by finding the center-of-contraction and the principal axes of the heart. We then fit the 3D data to superquadrics. The parameters of the modeling primitive are directly related to the global nonrigid motions, such as expansion or contraction and twisting deformation. The local movement of the left ventricle is estimated via a tensor based approach using the localized surfaces obtained through spherical harmonic interpolation of the residual distances between the data points and the fitted global superquadric surface. The estimation results of the left ventricle motion and deformation are then visualized through scientific visualization techniques which allow the estimated motions to be perceived interactively. We have produced a video tape containing some of the visualization results.

5. Representation and Navigation

There are two goals of our work in this area. First, we are interested in efficient representation of the surface shape information acquired as described in the previous sections. Second, we are interested in using the scene representation for path planning. Our work in both these areas has a common theme: the use of potential field as the underlying model.

5.1. Efficient Shape Representation using a Potential Field Model

We have worked on efficient derivation of the medial axis transform and the generalized cylinder representations of a two-dimensional region [17]. Instead of using the shortest distance to the region border, a potential field model is used for computational efficiency. The region border is assumed to be charged and the valleys of the resulting potential field are used to obtain prleiminary estimates of the axes for the two representations. The potential valleys are found by following force field thus avoiding two-dimensional search. First, the approach is presented to obtain the representation of a polygonal region. Closed form computation of the potential field is described using the equations of the border segments. The simple Newtonian potential is shown to be inadequate for deriving accurate representation. A higher order potential is defined which decays faster with distance than as inverse of distance. It is shown that as the potential order becomes arbitrarily large, the axes approach those computed using the shortest distance to the border. An algorithm is presented to efficiently compute the medial axis. This algorithm is then modified to obtain the generalized cylinder axis of a polygonal region. These algorithms for polygonal regions are used to perform a multiresolution, coarse-to-fine computation of the medial axis and generalized cylinder axes of arbitrary shaped regions.

5.2. Path Planning using a Potential Field Model

We have focused on the problem of path planning, i.e., deriving an efficient and collision free trajectory to move an object from a given source location/orientation to a give destination location/orientation through an environment whose occupancy map is given by the locations and shapes of the various objects. We have surveyed the state of the art approaches to autonomous path planning among obstacles in [10]. The problem is divided into two stages. First, a candidate topological path is selected. Second, the candidate path is cost-optimized to derive the final path and orientations of the moving object. In earlier work, we had used a simple potential field based model [37, 5] to represent the free space. We have now begun work with a more realistic, Newtonian potential function which also improves the computational efficiency. We have developed a Newtonian potential field based approach to path planning in which the design of an individual path segment is treated as a local problem in that the location and orientation of the object along the segment are determined only by the nearby obstacles [26, 25]. In particular, those parts of free space closely surrounded by obstacles where the object has to tightly maneuver through the obstacles are viewed as important components of the topological structure of the free space. These parts comprise bottlenecks for object motion. Between these bottleneck regions, the object motion is much more free. Thus, the top level planning determines an order in which the bottlenecks must be traversed for the object to move from source to destination. This decomposes the path planning problem into two sets of subproblems. Maneuvering through each bottleneck comprises one set of subproblems. Traversal of free space between the bottlenecks defines the second set of subproblems. The solutions of the second set of subproblems define path segments that link the solution paths which are obtained by solving the subproblems of the first set, thus yielding a global solution. For the 2D problems, free space bottlenecks are defined by the minimal distance links among polygonal obstacles, and object shape is represented by its skeleton. Both of these representations can be computed easily. The likelihood of collision between the object to be moved and the obstacles is measured through the Newtonian potential model wherein each object/obstacle region border is assumed to be charged. In the local problem for a given bottleneck, the exact location and orientation of the object, as the successive skeleton points enter the bottleneck, are determined so that the force and torque experienced by the object are minimized. Such repulsive forces and torques between polygonal regions can be computed analytically. Global planning is used to link the solutions of the local problems at adjacent bottlenecks into a global solution. Once the object exits a bottleneck, it is pulled by the next minimal distance link which is achieved by forcing the object to reduce its distance to the link. As the object is being pulled, the location and orientation of the object are changed to minimize the force and torque experienced. Thus, the object is forced to follow the path of minimal likelihood of collision between bottle-

necks. Because the distance to the next link decreases monotonically, a solution is guaranteed to be found if one exists. Due to such mixed use of geometry and potential field constraints, the local minima of potential fields do not present a problem.

References

[1] A.N. Choudhary, S. Das, N. Ahuja and J. Patel, High Speed Stereo, S. Prasad and R. L. Kashyap (Eds.), *Spectral Analysis in One or Two Dimensions*, Vedam Books International, 1990.

[2] A.N. Choudhary, J.H. Patel and N. Ahuja, Architecture and Performance Evaluation of NETRA, V.K. Prasanna Kumar (Ed.), *Parallel Architectures and Algorithms for Image Understanding*, Academic Press, 1991, 251-278.

[3] J. Weng, N. Ahuja and T. S. Huang, Motion and Structure from Point Correspondences With Error Estimation: Planar Surfaces, *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 39, No. 12, Dec. 1991, to appear.

[4] N. Ahuja and A. L. Abbott, Surfaces from Dynamic Stereo: Integrating Camera Vergence, Focus and Calibration with Stereo Surface Reconstruction, *IEEE Trans. Pattern Analysis and Machine Intelligence*, to appear.

[5] Y. Hwang and N. Ahuja, A Potential Field Approach to Path Planning, *IEEE Transactions on Robotics and Automation*, to appear.

[6] J. Weng, T. S. Huang and N. Ahuja, Motion and Structure from Line Correspondences: Closed-form Solution, Uniqueness and Optimization, *IEEE Trans. Pattern Analysis and Machine Intelligence*, to appear.

[7] X. Hu and N. Ahuja, Motion Estimation Under Orthographic Projection, *IEEE Trans. on Robotics and Automation*, to appear.

[8] J. Weng, N. Ahuja and T. S. Huang, Matching Two Perspective Views, *IEEE Trans. Pattern Analysis and Machine Intelligence*, to appear.

[9] J. Weng, N. Ahuja and T. S. Huang, Optimal Motion and Strcuture Estimation from Two Views, *IEEE Trans. Pattern Analysis and Machine Intelligence*, to appear.

[10] Y. Hwang and N. Ahuja, Gross Motion Plannig - A Survey, submitted to *ACM Computing Surveys*.

[11] A. N. Choudhary, J. H. Patel, and N. Ahuja, NETRA - A Parallel Architecture for Integrated Vision Systems I: Architecture and Organization, submitted to *IEEE Trans. Parallel and Distributed Processing*.

[12] A. N. Choudhary, J. H. Patel, and N. Ahuja, NETRA - A Parallel Architecture for Integrated Vision Systems II: Algorithms and Performance Evaluation, submitted to *IEEE Trans. Parallel and Distributed Processing*.

[13] C.R. Debrunner and N. Ahuja, Estimation of 3D Structure and Motion from Extended Point Trajectories, submitted to *Computer Vision, Graphics and Image Processing*.

[14] S. Sull and N. Ahuja, Integrated 3D recovery and Visualization of Flight Image Sequences, *these proceedings*.

[15] S. Sull and N. Ahuja, Integration of Multiple Features for 3D recovery and Visualization of Flight Image Sequences, submitted to *IEEE Trans. Pattern Analysis and Machine Intelligence*.

[16] X. Hu and N. Ahuja, Sufficient Conditions for Double or Unique Solution of Motion and Structure, summitted to *Computer Vision, Graphics and Image Processing*.

[17] N. Ahuja and R. Chuang, Shape Representation using a Generalized Potential Field Model, submitted to *IEEE Trans. Pattern Analysis and Machine Intelligence*.

[18] S. Das and N. Ahuja, Active Stereo Based Image Acquisition and Surface Reconstruction, *Proc. 5th IEEE Int. Symposium on Intelligent Control*, Philadelphia, Sep 5-7, 1990.

[19] S. Das and N. Ahuja, Integrated Multiresolution Image Acquisition and Surface Reconstruction from Active Stereo, *Proc. DARPA Image Understanding Workshop*, Pittsburgh, Sep 11-13, 1990, 418-422.

[20] E. Altman and N. Ahuja, A Dynamical Systems Approach to Integration in Stereo, *Proc. DARPA Image Understanding Workshop*, Pittsburgh, Sep 11-13, 1990, 423-427.

[21] X. Hu and N. Ahuja, A Robust Algorithm for Plane Motion Estimation, *International Conf. on Automation, Robotics and Computer Vision*, Singapore, Sep 18-21, 1990.

[22] S. Das and N. Ahuja, Multiresolution Image Acquisition and Surface Reconstruction, *Proc. 3rd Int. Conf. on Computer Vision*, Osaka, Japan, Dec 4-7, 1990, 485-488.

[23] A. L. Abbott and N. Ahuja, Active Surface Reconstruction by Integrating Focus, Vergence, Stereo and Camera Calibration, *Proc. 3rd Int. Conf. on Computer Vision*, Osaka, Japan, Dec 4-7, 1990, 489-492.

[24] S. Das and N. Ahuja, Dynamic Integration of Visual Cues for Position Estimation, *Proc. SPIE Conference*, Boston, 1990.

[25] R. Chuang and N. Ahuja, Path Planning using Newtonian Potential, *IEEE Conference on Robotics and Automation*, Sacramento, April 8-10, 1991, 558-563.

[26] R. Chuang and N. Ahuja, Path Planning using Newtonian Potential, submitted to *IEEE Trans. on Robotics and Automation.*

[27] X. Hu and N. Ahuja, Motion and Structure from Orthographic Images, *IEEE Conference on Acoustics, Speech and Signal Processing*, 1991.

[28] S. Sull and N. Ahuja, Estimation of Motion and Structure of Planar Surfaces from a Sequence of Monocular Images, *IEEE Conference on Computer Vision and Pattern Recognition*, 1991, 732-733.

[29] S. Sull and N. Ahuja, Segmentation and Estimation of Structure and Motion of Textured Piecewise Planar Surfaces, *IEEE Workshop on Visual Motion*, October 1991.

[30] J. Weng and N. Ahuja, Gradual Deviation of Visual World and Extended Topological Visual Mapping, submitted to *2nd ECCV*, 1992.

[31] X. Hu and N. Ahuja, Point Feature Matching, submitted to *IEEE Trans PAMI.*

[32] C. Debrunner and N. Ahuja, Motion and Structure Factorization and Segmentation of Long Multiple Motion Image Sequences, *these proceedings.*

[33] M. Tuceryan, A. K. Jain and N. Ahuja, Supervised Classification of Early Perceptual Structure in Dot Patterns, submitted to *ICPR.*

[34] C. Debrunner and N. Ahuja, Motion and Strucxture Factorization and Segmentation of Long Multiple Motion Image Sequences, submitted to *2nd ECCV*, 1992.

[35] E. Altman, Dynamical Systems Approach to Low Level Integration, *Proc. SPIE*, 1991, to appear.

[36] N. Ahuja and T. Huang, IU at UI: An Overview and an Example on Shape from Texture, *Proc. DARPA Image Understanding Workshop*, Boston, April 6-8, 1988, 222-253.

[37] N. Ahuja, IU at UI: An Overview of Research during 1988-90, *Proc. DARPA Image Understanding Workshop*, Pittsburgh, Sep 11-13, 1990, 134-140.

[38] M.S. Lew, K.W. Wong, and T. S. Huang, Applying concept learning to matching, submitted to *Int. Conf. on Patt. Recog.*, 1992.

[39] A.C. She, K.D. Hjelmsted, and T. S. Huang, Stereo camera measurements of displacements in a frame structure, submitted to *Int. Conf. on Patt. Recog.*, 1992.

[40] Yuncai Liu and Thomas S. Huang, Straight Edge Extraction and Matching, in *Proc. Close-Range Photogrammetry Meets Machine Vision*, International Society for Photogrammetry and Remote Sensing, ETH Zurich, Switzerland, Sep. 3-7, 1990.

[41] Mun K. Leung, Yuncai Liu and Thomas S. Huang, Experiment Results of 3D Motion Estimation Using Images of Outdoor Scenes, *Proc. DARPA Image Understanding Workshop*, Pittsburgh, Pennsylvania, Sep. 11-13, 1990, pp. 428-432.

[42] Mun K. Leung, Yuncai Liu and Thomas S. Huang, Estimating 3D Vehicle Motion in an Outdoor Scene from Monocular and Stereo Image Sequences, *IEEE Workshop on Visual Motion*, Oct. 7-10, 1991, Princeton, New Jersey.

[43] Yuncai Liu and Thomas S. Huang, Three-Dimensional Motion Interpreting from Real Scene Images Using Straight Line Correspondences, submitted to *Pattern Recognition*, April, 1991.

[44] Yuncai Liu and Thomas S. Huang, Determining Straight Line Correspondences from Intensity Images, in *Pattern Recognition*, vol. 24, no. 6, 1991.

[45] Yuncai Liu and Thomas S. Huang, A Model for vehicle-type Motion, submitted to *Second European Conf. Computer Vision*, Santa Margherita, Ligure, Italy, May 18-23, 1992.

[46] Yuncai Liu and Thomas S. Huang, Corner-Based Motion Analysis from Perspective Images of a Monocular Camera, submitted to *Artificial Intelligence*, Feb, 1991.

[47] Yuncai Liu and Thomas S. Huang, Vehicle-Type Motion Estimation from Multi-Frame Images: A New Model for Vehicle Motion, submitted to *IEEE Trans. PAMI*, April, 1991.

[48] Mun K. Leung, Yuncai Liu and Thomas S. Huang, Estimating 3D Vehicle Motion in an Outdoor Scene from Monocular and Stereo Image Sequences, submitted to *Comput. Vision Graphics Image Processing*, March, 1991.

[49] Yuncai Liu and Thomas S. Huang, Estimating 3D Vehicle Motion in an Outdoor Scene Using Line Correspondences, to appear in *International Conf. Manufacturing Automation*, Hong Kong, August, 1992.

[50] Yuncai Liu and Thomas S. Huang, Error Analysis for Linear Motion Estimation Using Straight Line Correspondences, to appear in *International Conf. Manufacturing Automation*, Hong Kong, August, 1992.

[51] Yuncai Liu and Thomas S. Huang, Comparison of Different Methods of 3D Motion Estimation Using real Scene Image Sequence, to be submitted to *2nd Singapore International Conference on Image Processing*, IEEE Singapore Section, Singapore, 7-11 Sep. 1992.

[52] M. K. Leung and T. S. Huang, Detecting Wheels in Vehicles, to appear in *Pattern Recognition*, 1992.

[53] T. S. Huang, S. Reddy and K. Aizawa, Modeling Analysis and Visualization of Human Facial Motion, *Proc. SPIE Conf. on Visual Communication and Image Processing*, Nov. 11-15, 1991, Boston, Mass.

[54] Jialin Zhong, Juyang Weng, Thomas S. Huang, Vector field interpolation in fluid flow, Digital Signal Processing, *1991 Int. Conf. on DSP*, Florence, Italy, Sept. 4-6, 1991

[55] Jialin Zhong, Juyang Weng, Thomas S. Huang, Vector field interpolation using robust statistics, *Proc SPIE Conf. on Curves and Surfaces in Computer Vision and Graphics II*, Boston, Nov. 11-15, 1991.

[56] Jialin Zhong, Juyang Weng, Thomas S. Huang, Robust and Physically-Constrained Interpolation of Fluid Flow Fields, *IEEE Conf. on ASSP*, San Francisco, March 22-26, 1992.

[57] C.W. Chen and T.S. Huang, Analysis of left ventricle global deformation based on dynamic CT data, *International Conf. Pattern Recognition* (submitted), August 1992, The Hague, The Netherlands.

[58] C.W. Chen and T.S. Huang, Modeling, Analysis and Visualization of Left Ventricle Shape and Motion by hierarchical decomposition, *IEEE Trans. Pattern Anal. Machine Intell.*, (submitted).

[59] C.W. Chen and T.S. Huang, Left ventricle motion analysis by hierarchical decomposition, *Proc. International Conf. on Acoustics, Speech, Signal Processing '92*, March 1992, San Francisco, CA.

[60] C.W. Chen and T.S. Huang,Surface modeling in heart motion analysis, *Proc. SPIE Conf. on Curves and Surfaces in Computer Vision and Graphics II*, November 1991, Boston, MA.

[61] C.W. Chen, T.S. Huang and Y.C. Chen, Model based estimation of left ventricle motion, *Proc. International Conf. on Acoustics, Speech, Signal Processing '91*, pp 2481-2484, May 1991, Toronto, Canada.

[62] C.W. Chen, T.S. Huang and M. Arrott, Analysis and visualization of heart motion, *Proc. SPIE Conf. on Biomedical Image Processing II*, Vol. 1450, pp 231-242, Feb. 1991, San Jose, California.

[63] C.W. Chen and T.S. Huang, Nonrigid object motion and deformation estimation from three-dimensional data, *International Journal of Imaging Systems and Technology*, Vol. 2, pp 385-394, 1990.

[64] C.W. Chen and T.S. Huang, Epicardial motion and deformation estimation from coronary artery bifurcation points, *Proc. Third International Conf. on Computer Vision*, pp 456-459, Dec. 1990, Osaka, Japan.

[65] C.W. Chen and T.S. Huang, A new algorithm for motion/structure estimation using error probability, in *Time-varying Image Processing and Moving Object Recognition, 2*, V. Cappellini ed., Elsevier Science Publishers B.V. Amsterdam, pp 250-257, 1990.

[66] Yuncai Liu and Thomas Huang, Vehicle-type Motion Modeling and Analysis, *these proceedings.*

[67] R. J. Qian and Thomas Huang, Motion Analysis of Articulated Objects with Applications to Human Ambulatory Patterns, *these proceedings.*

[68] R. J. Qian and T. S. Huang, Motion Analysis of Articulated Objects, submitted to *ICPR*, 1992.

An Active Observer

Ruzena Bajcsy*
GRASP Laboratory
Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA 19104

1 Abstract

In this paper we present a framework for research into the development of an **Active Observer**. The components of such an observer are the low and intermediate visual processing modules. Some of these modules have been adapted from the community and some have been investigated in the GRASP laboratory, most notably modules for the understanding of surface reflections via color and multiple views and for the segmentation of three dimensional images into first or second order surfaces via superquadric/parametric volumetric models. However the key problem in Active Observer research is the control structure of its behavior based on the task and the situation. This control structure is modeled by a formalism called Discrete Events Dynamic Systems (DEDS).

2 Introduction

We are interested in the development of an Active Observer. An Active Observer is an agent which has capabilities to observe scenes, objects, situations and deliver the observed information to human, manipulatory, and mobile agents. Naturally there are more questions than answers. We shall list a few which are of particular interest to us. What are the components/modules that such an observer must have? How are these components interconnected, i.e. what is the architecture of such an agent? Some of the modules correspond to certain visual cues. We take as a given that our observer has several such cues. In that case, the subsequent question is how are the results from these cues integrated? When are they invoked? How is the selection process conducted/guided? Which cue is employed and when? Finally, what kind of information/messages is delivered by the observer to other agents?

Towards this end, for the last two years we have concentrated on the development of theoretical and experimental understanding of some of the cues/components, some cues' integration and selection, and control strategies for observation capability. In particular, in cue development we have tried to understand surface reflections by color and multiple views. An important finding of this work, which will be described in detail in Section 2, is that multiple view points provide useful information for discriminating between specular and Lambertian reflections both from dielectrics and from metals. In Section 3, we shall describe a system for the segmentation of a three dimensional scene into components that can be modeled by superquadric parametric fit. This system uses, in cooperation, surface segmentation, contour segmentation and gross volumetric segmentation in order to arrive at the proper result. The scenes are of moderate complexity (up to 10 parts), but no other assumptions are made about objects or their parts. This work points to the common fact that one module or cue or approach cannot handle the perceptual variety of the data that the real world, even in moderate complexity, represents. Multiple cues are necessary and hence a great deal of thought has to go into the integration policy and control structure. In Section 4, we present a formal model of an observer agent. This model is based on the theory of Discrete Event Dynamic Systems (DEDS), which allows us to unequivocally predict the observation capabilities of an observer. In order for this to occur, the observer must know the discrete events of the task. So far this is done by the designer. Finally, in Section 5 we show the recent development of a CCD chip (the Retina) with space variant resolution. Details are described in this section.

3 Understanding of Reflection Properties Using Color and Multiple Views

Recently there has been a growing interest in the detection of specularity in both basic and applied computer vision research. In general, the detection of specularities from a single gray-level image is a physically underconstrained problem, and more information needs to be collected in physically sensible ways to solve the problem. Successful development of an algorithm for image data collection and interpretation necessarily depends on physical models that describe how surfaces appear according to the illumination and reflectance properties and sensor characteristics. Recently the computer vision field has increasingly incorporated methodologies derived from physical principles of image formation and

*Acknowledgement: Navy Grant N0014-88-K-0630, AFOSR Grants 88-0244, AFOSR 88-0296; Army/DAAL 03-89-C-0031PRI; NSF Grants CISE/CDA 88-22719, IRI 89-06770; and Dupont Corporation

sensing [7]. So far there have been three types of approaches to solving the problem of specularity detection through the collection of more images: (1) with different light directions, (2) with different sensor polarization angles, and (3) with different color sensors.

The photometric-stereo-type approaches consider the specular and Lambertian reflectance properties for obtaining object shape using more than two light directions [4] [9] [11]. Since the direction and the degree of the collimation of the illumination need to be strictly controlled, application of the approach is restricted to dark-room environments. The polarization method analyzes the polarization of reflected light and detects specularities from dielectrics and metals [12]. The polarization approach places some restrictions on the incident illumination direction with respect to surface orientation.

The dichromatic model [10] proposed by Shafer has been the key model to the recent specularity detection algorithms using color [8] [5] [6] [3]. The basic limitation of the color algorithms is that objects must be only colored dielectrics to use the dichromatic model. For color image segmentation, it is usually assumed that object surface reflectance is spatially piecewise uniform in color and that scene illumination is singly colored. We have previously developed a color image segmentation algorithm for the separation of diffuse, as well as sharp, specularities and inter-reflections from Lambertian reflections [3].

Our recent research has focused on the development of some specularity detection or separation methods that only require modification of sensors but not any modification of environments. In other words, they are methods that are active in modifying sensors but passive in modifying environments. There are two kinds of modification of environments: relocation and re-orientation of objects by robot manipulation, and illumination change. The prime example of the illumination change is the light switching for the photometric-stereo-type methods. Since illumination lighting needs to be strictly controlled, the photometric-stereo-type approaches are applicable only for inspection in dark rooms.

Strict illumination control is not always possible in investigating surface reflection properties in many general environments. Examples include outdoor inspection, indoor or outdoor navigation, and exploratory environments. Even for indoor inspection, a well controlled dark room is not always available.

For general environments without strict illumination control, only sensors are controllable, and color and polarization can be the possible cues. Another possibility is to move the observer, which has not been used for investigating reflection properties in computer vision. The idea of moving the observer was directly motivated by the concept of active vision [2]. For low-level vision problems of shape or structure, it has been demonstrated that many ill-posed problems become well-posed if more information is collected by active sensors [1]. Although the paradigms for shape or structure based on feature correspondence cannot be directly applied to the study of reflectance properties, the idea of a moving observer motivated the investigation of new principles by physical modeling in obtaining more information.

In this paper, we suggest the use of multiple views for the detection of specularity by introducing two algorithms. The first algorithm, called spectral differencing, uses color information from a small number of multiple views. The second algorithm is called view sampling. Using many views of gray-level images collected in wide angle, the view sampling reconstructs object structure and detects specularities. An important principle used for the algorithms is the Lambertian consistency, which is the well-known fact that the Lambertian reflection does not change its brightness and spectral content depending on viewing directions, but the specular reflection or the mixture of Lambertian and specular reflections can change.

A problem associated with the use of multiple views with color is what kind of extra spectral information can be obtained by moving a color camera without considering object geometry. If there is any, it may alleviate the limiting assumptions imposed on the object and illumination domain for the color segmentation approaches, and provide higher confidence in detecting specularities.

The spectral differencing algorithm is based on the observation that any presence of specular reflections can be inferred by the difference in the distribution of pixel colors between two color images. According to the Lambertian consistency, the color distribution of pixels from only Lambertian reflections should be consistent regardless of view points. On the other hand, specularities or the mixture of specular and Lambertian reflections can change the distribution of pixel colors between two views.

The spectral differencing algorithm does not require any assistance from image segmentation and geometrical manipulation. Since the algorithm does not rely on the segmentation and the dichromatic model, it is applicable to dielectric objects with nonuniform reflectance and metals under multiply colored illumination. Figures 1 and 2 show two dielectric objects with variation in reflectance and a metallic object in neutral reflectance color. Two fluorescent light tubes and a tungsten light bulb are used for illumination and there are inter-reflections from the walls. MSD($0 \leftarrow 1$) shows the regions of new color distribution in view 0 compared to view 1, and MSD($1 \leftarrow 0$) the regions of new color distribution in view 1 compared to view 0. Under multiply colored and extended illumination, it can be seen that most of the specularities are detected by the spectral differencing.

Another approach we introduce is to obtain reflection properties using only multiple views without any color information. With densely sampled views in wide angle and with known viewing directions, the view sampling algorithm reconstructs object structure as well as detects specularities from Lambertian reflections. The view sampling algorithm is applicable to dielectrics and metals.

If object structure is reconstructed assuming the Lambertian consistency for both Lambertian and specular reflections, the structure reconstructed from the specular reflections would not in general represent the real object

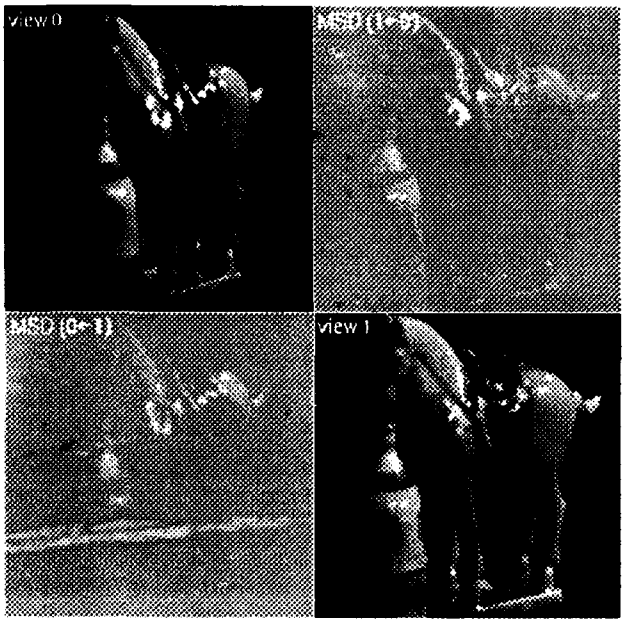


Figure 1: Spectral differencing

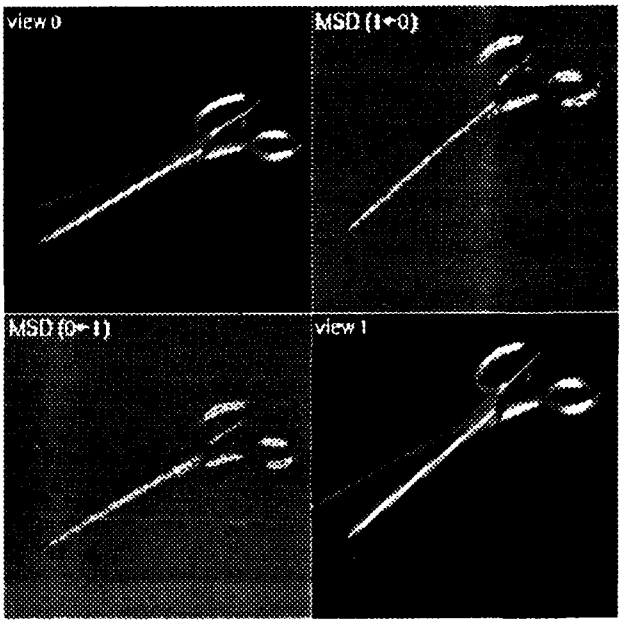


Figure 2: Spectral differencing

surface, while the one reconstructed from the Lambertian reflections does. By examining the differently reconstructed object structures from specular and Lambertian reflections, we can identify the reflection types and the real object structure.

We adopted an algorithm for computerized tomography through photometric modeling for the reconstruction of object structure. Figure 3 shows the camera control scheme and Figure 4 (a) shows 4 out of 30 view samples of a gray dielectric object from different view points. Figure 4 (c) and (d) show the reconstructed structures at the cross sections 1 and 2 illustrated in Figure 4 (b), respectively. As shown in Figure 4 (c) and (d), the structure reconstructed from specularities at the cross section 2 is different from the real object surface reconstructed by Lambertian reflections.

The future direction of our studies is the integration of many cues in the light of active vision [2]. Active vision involves not only the modeling of physical sensing and data processing for vision modules (local model), but also the control of the modules (global model). Global models characterize the overall performance and make predictions on how the individual modules will interact, which in turn determines how intermediate results are combined. It is the global model that analyzes and combines the information from many visual cues to assign stable descriptors. For more stable descriptions of reflection properties in more general environments, it is desirable to extract extra information from a synergistic combination of multiple cues. The spectral differencing algorithm demonstrates the synergy from the combination of color and multiple views. There are also potentials for extra information from the combination of color, polarization and multiple views.

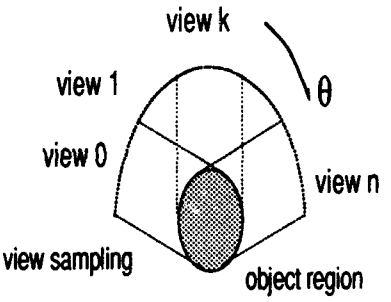


Figure 3: View sampling

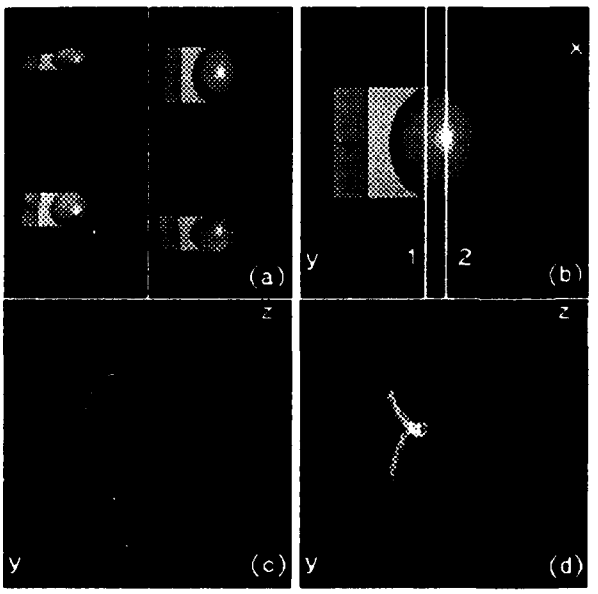


Figure 4: View sampling

4 Surface and Volumetric Segmentation of Complex 3-D Objects Using Parametric Shape Models

The problem of part definition, description, and decomposition is central to shape recognition systems. In this paper we present an integrated framework for segmenting dense range data of complex 3-D scenes into their constituent parts in terms of surface (bi-quadrics) and volumetric (superquadrics) primitives, without a priori domain knowledge or stored models. Our objective is to *recover* a structured description of complex 3-D objects, guided entirely by the geometric properties of the shape models. The resulting decomposition into parts is very useful for the high-level processes, which can attach domain specific labels to the parts, and reason at a level where the visual input is structured in terms of geometric primitives, rather than cope with the difficulties of low-level vision and a huge amount of unstructured data.

Since the shapes have to be recovered from raw data, it is not possible to invoke complex models (models with hundreds of degrees of freedom) straight away. It is, however, feasible and perceptually less ambiguous to use simpler but powerful models that can capture the local and global properties of the object shapes, and provide a first approximation to the more complex models. With computability, simplicity, and the utility of the shape representation as our major concerns, we use bi-quadrics and superquadrics as our surface and volumetric models respectively. We develop SUPERSEG (SUPERquadric SEGmentation), a control structure to effectively carry out the decomposition of complex objects in range images, and address the numerous issues encountered in a data-driven bottom-up approach [13; 14; 15].

The SUPERSEG system 5 has five major components: namely, the bi-quadric surface segmentation module; the module for extracting surface properties and adjacency relationships; the superquadric model recovery module; the residual generation and analysis module; and the control module for superquadric-based segmentation.

4.1 Surface Segmentation: Bi-quadric Models

The surface segmentation is performed by a novel local-to-global iterative regression approach of searching for the best piecewise description of the data in terms of bi-quadric models [16; 17]. The model-recovery module consists of independently extrapolating all the seed-regions and fitting the models using the least-squares regression method. The region-growing is controlled by a *compatibility-constraint*, whose value depends on the noise due to sensor and quantization, as well as the allowed tolerance between the shapes of the model and underlying data. Seed-regions are placed in a grid-pattern all over the image, and allowed to grow until they are either completely grown or rejected by the model-selection procedure (which maximizes a linear benefit-cost function). Instead of first growing all the regions and then invoking the model-selection procedure (Recover-then-select), the model-recovery and model-selection processes are dynamically combined (Recover-and-select) to achieve a computationally feasible and robust method capable of rejecting outliers and determining its domain of applicability.

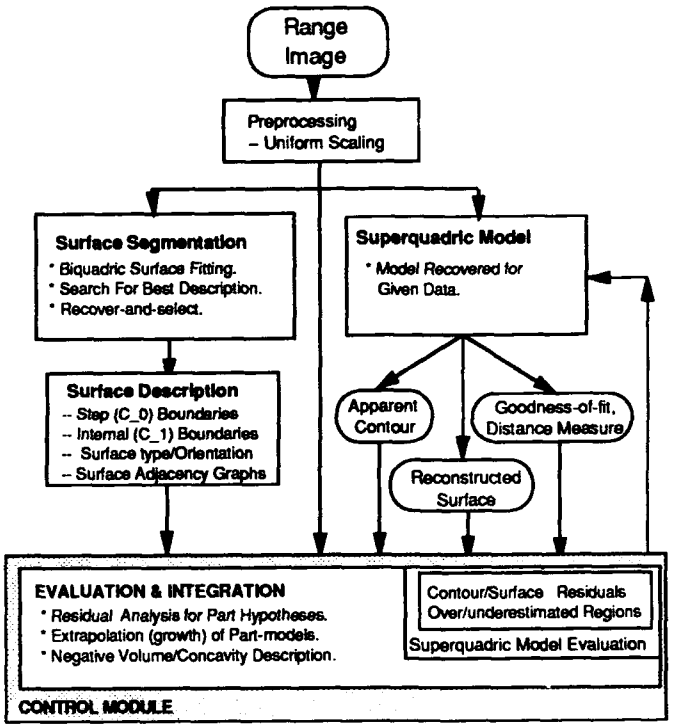


Figure 5: The SUPERSEG system: A framework for surface and volumetric segmentation.

4.1.1 Refining Surface Segmentation & Extracting Surface properties

The bi-quadric segmentation achieved by the above procedure needs refinement before it can be used as an intermediate segmentation by superquadric-based volume segmentation. Also, the coefficients of the second-order surfaces have information about orientation and surface-type (convex or concave) inherent in them. The orientation information is tremendously useful in aligning the major axis of cylindrical superquadric models. Further, due to the compatibility-constraint, regions intersecting to form surface normal discontinuities (C_1) overlap in the vicinity of the discontinuity, thereby localizing it. We developed a systematic method for tracing the biquadric intersection curve, which is used to refine the segmentation as well as to localize the discontinuities (edges) and to characterize them as convex or concave. In addition, a surface adjacency graph (SAG) is constructed with surface patches as nodes and discontinuity-type as edges between them. The information extracted from the bi-quadric patches is used to generate and test hypotheses by the volumetric segmentation module.

4.2 Superquadrics: Volumetric Part-Models

Superquadric models are convex part-models (except the bent models) that can be recovered for a given set of 3-D points by minimizing a function based on the modified implicit inside-outside superquadric function [18; 19;

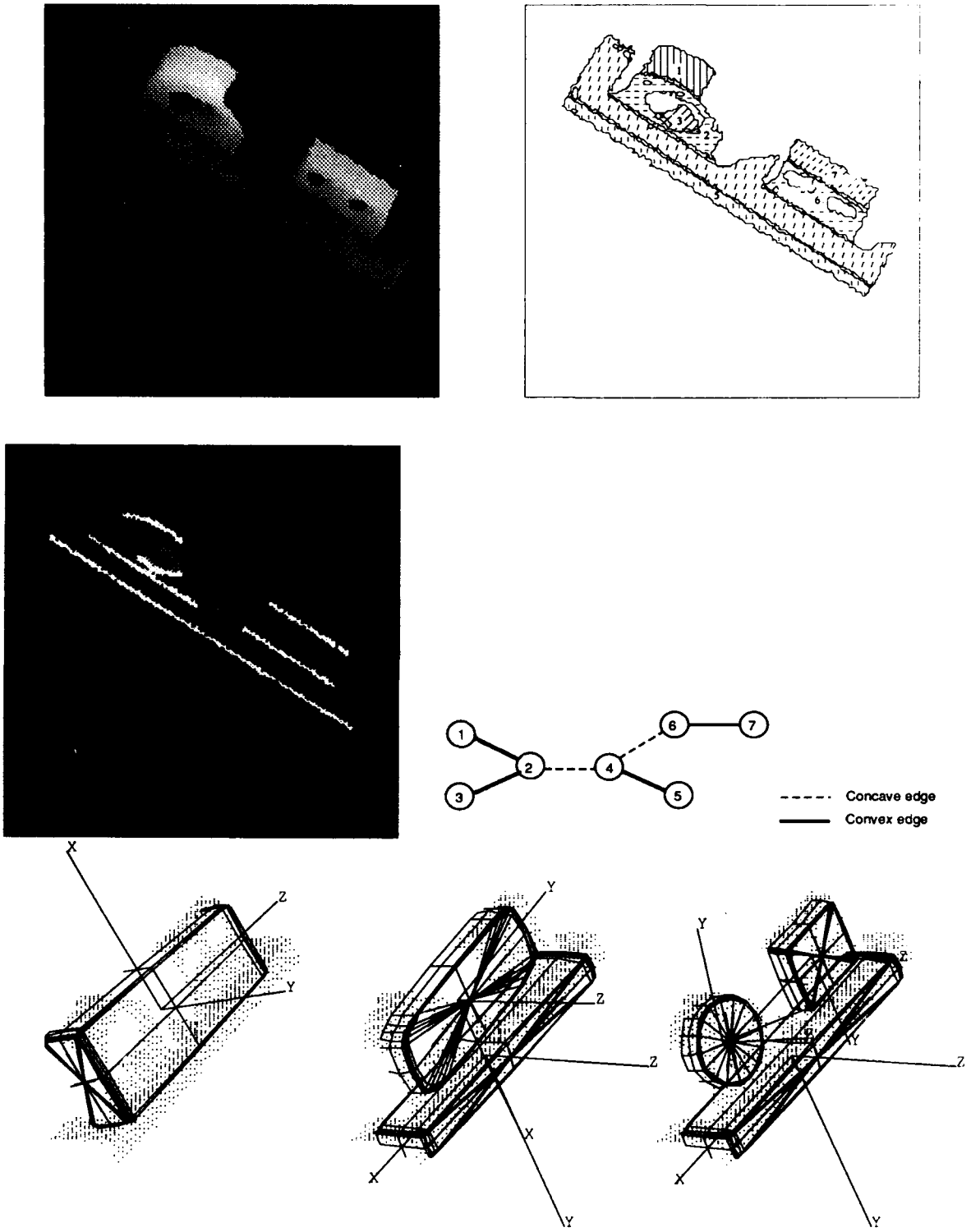


Figure 6: **The NIST object:** Top: The range image and its bi-quadric surface segmentation. Center: the C_1 (surface normal) edges marked at the overlapping parts of the surfaces. Following a procedure similar to the intersection cleaning, the edges are marked as convex or concave and a surface adjacency graph (SAG) is constructed. Bottom: The three iterations of the global-to-local procedure to extract the part-structure.

15]. This formulation enforces a minimum volume constraint as well as a surface constraint, but is incapable of decomposing the data set if no appropriate convex model can be found in the model vocabulary. Thus, the superquadric model recovery module is adequate only for recovering an optimal model (if oriented correctly) given a data set, but not for segmenting it. To decide whether a recovered model is adequate for the given data set, we have developed an exhaustive set of criteria comprised of qualitative and quantitative measures. Quantitative measures are the normalized global deviation of the model from data. The deviation can be the inside-outside function value, or can be measured along the direction of the viewpoint (Z-residuals for a range scanner), or along the direction of the minimum distance of a point from the model (Euclidean distance). The qualitative measures are the 'local' residuals characterized by the clusters of 3-D points that are either inside the model, or on the model, or outside the model. Both qualitative and quantitative measures are necessary for complete evaluation of a recovered model.

4.3 Volumetric Segmentation: The Control Strategy

In view of the fact that volumetric models don't have good surface support (as opposed to bi-quadric models), they cannot be recovered by following exclusively the extrapolation method (local-to-global) used by bi-quadrics. In order to obtain an optimal piecewise-convex volumetric segmentation, it is necessary to proceed global-to-local, where data is decomposed only if the global model is inadequate. This allows controlled residual-driven decomposition of 3-D data, as also introduction of an objective evaluation criteria for an acceptable description. However, the global-to-local method can be aided by the bi-quadric segmentation in forming hypotheses about convex combination of surfaces, which although is not true in general (an L shape for example), can significantly reduce the computational overhead if true for a particular part. Previous researchers have assumed that a 1-to-1 mapping exists between surface patches and superquadric models, which is also not true in general. But it does provide a planarity check for the patches, as well as the orientation and shape of the individual patches in 3-space.

Thus, a strategy that combines the bi-quadric information with the global-to-local residual-driven method is most effective in recursively segmenting the scene to derive the part-structure [13]. A set of acceptance criteria based on the quantitative and qualitative measures provide the objective evaluation of intermediate descriptions, and decide whether to terminate the procedure, or selectively refine the segmentation, or generate negative volume description. The control module generates hypotheses about superquadric models at clusters of underestimated data and performs controlled extrapolation of part-models by shrinking the global model. The recursive splitting of data results in a hierarchical part-structure comprising of global and local models. The results of complete processing of the range image of a machined object (from NIST) is shown in Figure 6.

We have tested the SUPERSEG system on real range images of scenes of varying complexity, including objects with occluding parts, and scenes where surface segmentation is not sufficient to guide the volumetric segmentation. Some of the applications of our approach include, data reduction, 3-D object recognition, geometric modeling, automatic model generation, object manipulation, qualitative vision, and active vision.

5 A Framework for Visual Observation

In this work we establish a framework for the general problem of observation, which may be applied to different kinds of visual tasks. We define "intelligent" high-level control mechanisms for the observer in order to achieve efficiency in recognizing different processes within a specific dynamic system. The intelligent observer is able to recognize the visual tasks, understands the meaning of the scene evolution and successfully reports on the current visual state. It is obvious that there is a need for high-level interpretation of actions within the environment and to have guarantees for observation capabilities and stability within the viewing mechanism. The framework is a *predictable* one that satisfies the following general requirements:

- Recognizes visual tasks and events.
- Repositions itself adaptively and intelligently.
- Operates in real time.
- Asserts and reports on *distinct* and *discrete* visual states.
- Utilizes the *continuous* parametric evolution of the visual system.
- Accommodates visual uncertainties.

We concentrate on observing a manipulation process in order to illustrate the ideas and motive behind our framework. The process of observing a robot hand manipulating an object is very crucial for many robotic and manufacturing tasks. It is important to know in an automated manufacturing environment whether the robot hand is doing the correct sequence of operations on an object (or more than one object). It might be a fact that the workspace of the robotic manipulator cannot be accessed by humans, as in the case of some space applications or some areas within a nuclear plant, for example. In such a case, having another robot "look" at the process is a very good option. Thus, the observation process can be thought of as a stage in a closed-loop fully automated system where there are robots who perform the required manipulation task and some other robots who observe them and correct their actions when something goes wrong. Typical manipulation processes include grasping, pushing, pulling, lifting, squeezing, screwing and unscrewing. In this work, we address the problem of observing a single hand manipulating a single object and recognizing what the hand is doing. No feedback will be supplied to the manipulating robot to correct its actions. We divide the problem into three major components. First, we identify a high-level framework for the visual states. Next, we define the events that cause

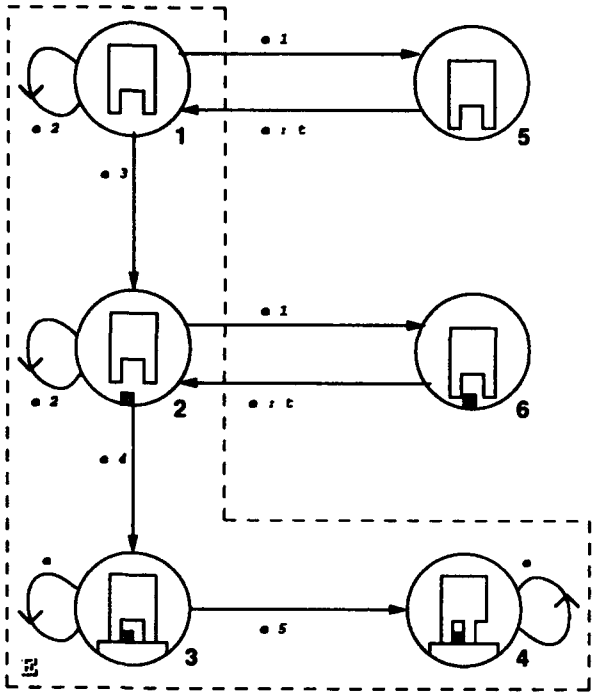


Figure 7: A Model for a Grasping Task

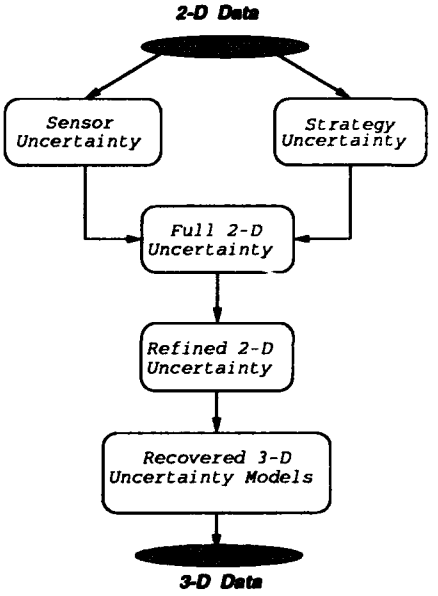


Figure 8: Propagation of Uncertainty

state transitions. Finally, we utilize visual uncertainties to assert the state of the system.

5.1 State Space Modeling

We use a discrete event dynamic system as a high-level structuring skeleton to model the visual manipulation system. Discrete event dynamic systems (DEDS) are dynamic systems (typically asynchronous) in which state transitions are triggered by the occurrence of discrete events in the system. Our formulation uses the knowledge about the system and the different actions in order to solve the observer problem in an efficient, stable and practical way. The model incorporates different hand/object relationships and the possible errors in the manipulation actions. It also uses different tracking mechanisms so that the observer can keep track of the workspace of the manipulating robot. A framework is developed for the hand/object interaction over time and a stabilizing observer is constructed. The construction process utilizes a task-dependent coarse quantization of the manipulation actions in order to attain an active, adaptive and goal-directed sensing mechanism. An example of a DEDS automaton for a simple grasping task is shown in Figure 7.

5.2 Event Identification

Low-level modules are developed for recognizing the "events" that cause state transitions within the dynamic manipulation system. To be able to observe how the events evolve over time, we must be able to identify how the hand moves and how the hand/object physical relationship evolves over time. We use a mix of 2-D and 3-D modules to recover a set of parameters that define the *continuous* parametric evolution of the scene under observation. Three dimensional evolution of the hand motion is recovered by tracking a set of features and two-dimensional cues to the number of objects and their relative location; two dimensional motion with respect to the manipulating hand is recovered in real-time. The recovered events are then used to assert state transitions within the DEDS automata. We also recover uncertainties associated with the visual event recovery and utilize them for navigating the observer automata.

5.3 Utilizing Uncertainties

This work examines closely the possibilities for errors, mistakes and uncertainties in the visual manipulation system, observer construction process and event identification mechanisms. We divide the problem into a number of major *levels* for developing uncertainty models in the observation process. The propagation of uncertainty is shown in Figure 8.

The *sensor* level models deal with the problems in mapping 3-D features to pixel coordinates and the errors incurred in that process. We identify these uncertainties and suggest a framework for modeling them. The next level is the *extraction strategy* level, in which we develop models for the possibility of errors in the low-level image processing modules used for identifying features that are to be used in computing the 2-D evolution of the scene under consideration. In the following level, we utilize the geometric and mechanical properties of the hand and/or object to reject unrealistic estimates for 2-D movements that might have been obtained from the first two levels. We transform the 2-D uncertainty models into 3-D uncertainty models for the structure and motion of the entire scene. The next level uses the equations that

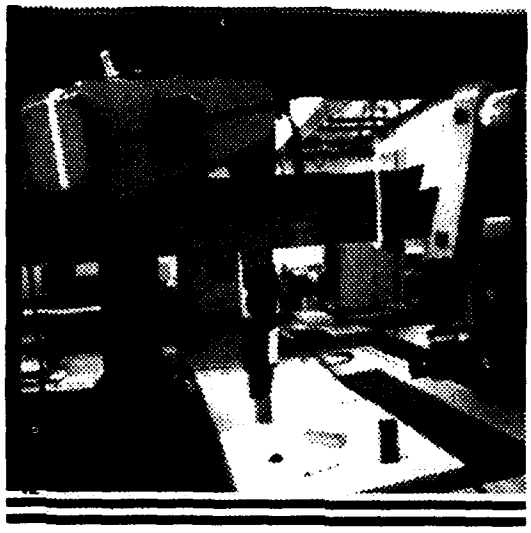


Figure 9: Experimental Setting

govern the 2-D to 3-D relationship to perform the conversion. We then reject the improbable 3-D uncertainty models for motion and structure estimates by using the existing information about the geometric and mechanical properties of the moving components in the scene. The highest level is the DEDS formulation with uncertainties, in which state transitions and event identification is asserted according to the 3-D models of uncertainty that were developed in the previous levels, and error recovery is performed according to the ordering of the recovered distributions.

5.4 Conclusions

The approach used can be considered as a framework for a variety of visual tasks, as it lends itself to be a practical and feasible solution that uses existing information in a robust and modular fashion. The work examines closely the possibilities for errors and uncertainties in the manipulation system, observer construction process and event identification mechanisms. Ambiguities are allowed to develop and are resolved after finite time; recovery mechanisms are devised too. Details of the observer system can be found in [20; 21; 22; 23]. Theoretical and experimental aspects of the work support adopting the framework as a new basis for performing task-oriented recognition, inspection and observation of visual phenomena. The observer and manipulating robots experimental setup is shown in Figure 9.

6 Spatio-Variant Sensing

Traditional imaging for robotics vision has relied almost exclusively on common commercial imagers, notably television format sensors. Their advantages are clear: the cameras are inexpensive and readily available, and the sampling of the data is on a "natural" cartesian (x,y) grid. These sensors have placed enormous demands, however, on processing architectures. The problem is not only that image analysis is an ill-defined task in the real world, but that we have only very expensive machines that can begin to process the data.

Over the last seven years an international team, led by Van der Spiegel at the University of Pennsylvania, Sandini at DIST in Italy, and Claeys at IMEC in Belgium, designed, built, and tested a new imaging chip called the Retina [24]. The new camera serves as the foundation to a new approach to robotics vision. We shift the focus at the systems level from gathering better data and designing machines to analyze it to gathering data for the computing resources that exist. The result is a prototype sensor that reduces the computational complexity of the problem by three orders of magnitude and, if scaled to commercial cameras, by six orders [25].

The Retina attempts to model the gross characteristics of the primate visual system in a mathematically elegant way. The computational savings arise from the same mechanism the eye uses, namely, to maintain one area of high resolution on the focal plane and to drop the resolution elsewhere. The mathematical expression of this is a log-polar mapping. That mapping transforms a polar data space, where a point P has the polar coordinates $(r, theta)$, by taking the logarithm of the expression for the point:

$$P = re^{i\theta} \longrightarrow P' = ln(r) + i\theta = u + iv$$

This mapping has the useful property of separating rotations (changes in *theta*) from magnifications (changes in r). If the sensor has a uniform sampling grid in u (ln (r)), then the spatial grid in r will exponentially grow as distance from the center grows. This models the growth of the receptive fields in primate retinas.

The Retina layout in Figure 10 implements this mapping by sampling in $(r, theta)$ at points matching a uniform (u, v) grid. The sensor clearly has rotational symmetry and exponentially decreasing resolution. The circular section contains only 1920 pixels (30 circles of 64 pixels/circle); at the center is a dense rectangular grid of 102 additional photosites [26]. The cells grow fast: the outermost circle is over ten times as wide as the innermost. This leads directly to the small pixel count.

The chip, with its custom driving electronics, is now working at the GRASP laboratory [27] and is producing good pictures as shown in Figure 11.

Clearly visible in the data space is the large magnification of the inner circles. The outer section provides much poorer data, with pixels widely spaced and averaging the incident light over a larger area. Still they do not provide useless information.

The nature of the information has changed, however. No longer do we get high quality data across the focal plane. Indeed, we assume from the start that we do not try to build a model of the world in one step. Instead, we use the periphery to guide our attention - where we point the camera. Implicit here is the idea of an active observer. The Retina, just sitting on a bench waiting for an object to enter its high-resolution spot, is useless. We must actively build the world by moving the camera, using the periphery to suggest candidates for attention.

The cost of using this sensor might be considered high. The new data space will require rewriting or adapting

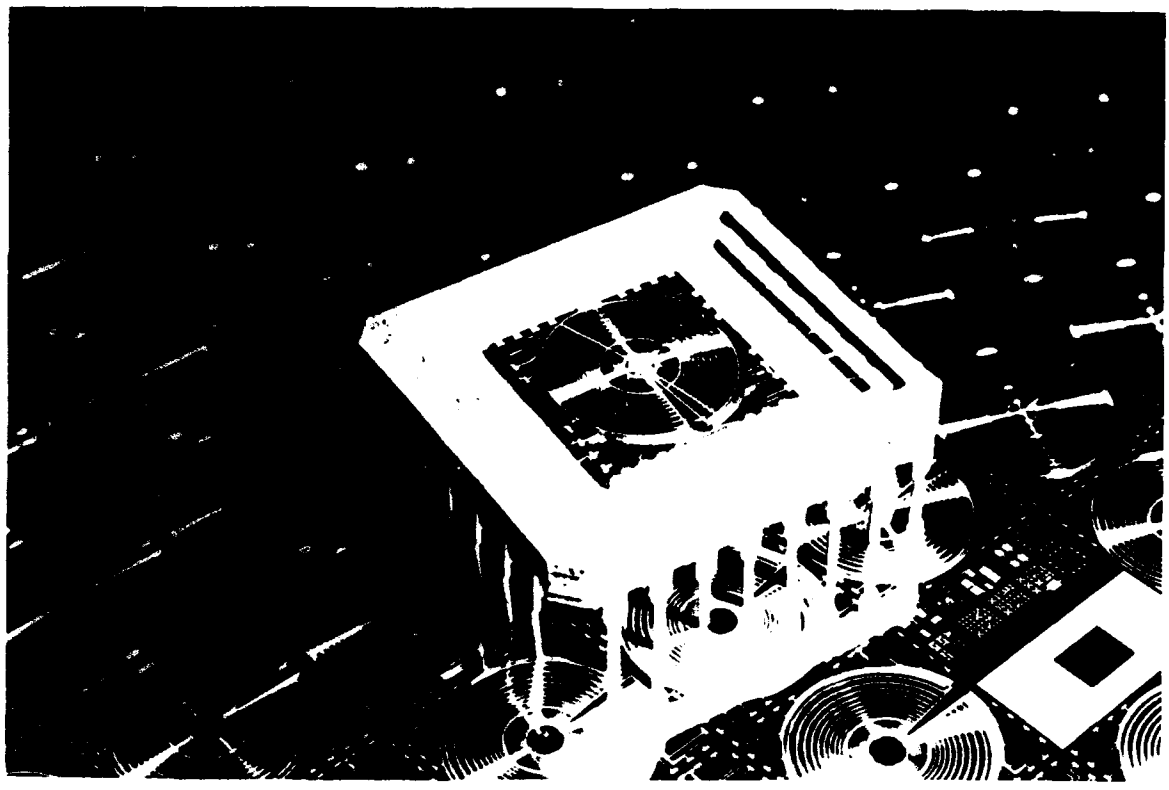


Figure 10: The Retina CCD Imager

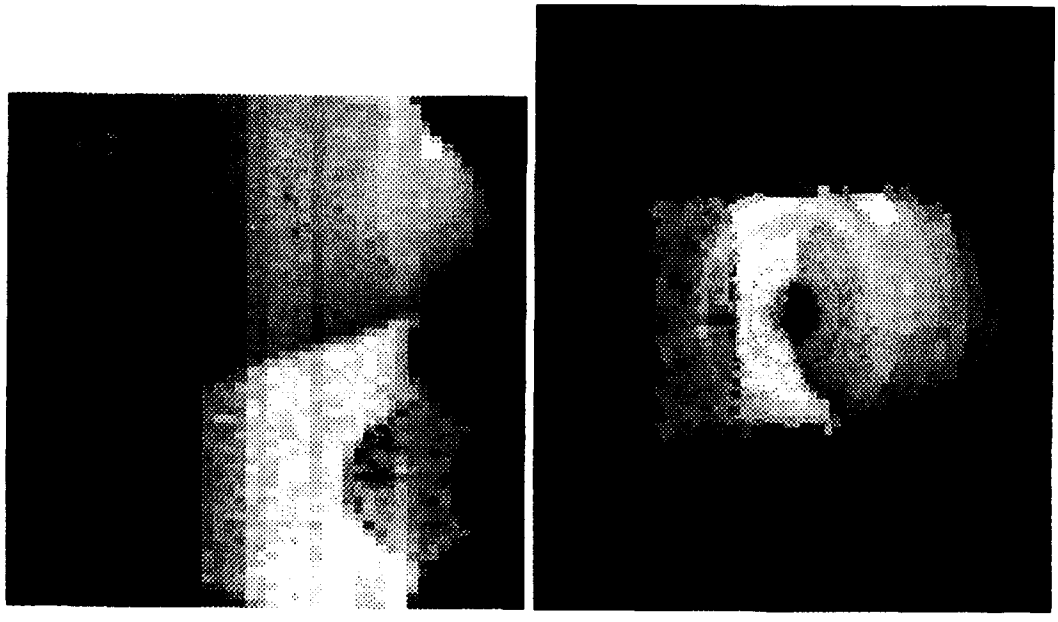


Figure 11: Picture of a mouse from the camera, centered between the buttons (to the left) and ball. The picture on the left is in the mapped plane: the vertical axis is v (v, the angle of the point, increases moving down the axis) and the horizontal is u (u, the log of the radial distance of the point, increases to the right). The triangle at the upper left of the image is data remapped back onto a cartesian grid.

all our tools for the cartesian plane; this is the primary cost outside the hardware development. The advantages, however, suggest profit. The Retina has some one hundred times fewer pixels than a standard television camera, which drastically reduces the computational burden of analysis, bringing it within the abilities of modern machines. The gains also include the rich mathematical structure of the mapping. That structure simplifies pattern matching by making rotations and magnifications linear shifts in the data space, and speeds time-to-impact measurements by looking only at a radial flow. Some distortions introduced by the mapping, such as translational variance (which is linear translations becoming curves in the data space) also disappear in an active observer, where for example attention and tracking automatically compensate for linear motion.

Since the sensor began working this summer, our focus at the GRASP laboratory has been redeveloping traditional image processing tools. Our work has looked at edge detection in the new data space, detecting lines using a Hough algorithm, calculating the centroid of an object, and measuring time-to-impact. Each of these areas requires an analysis of their mathematical basis under the log mapping and coding the results on real images. All algorithms must further be computationally simple to work in a real-time environment.

This integration of sensor and computer is now the fundamental area of research involving the Retina at Penn. That the Retina works proves the concept of the hardware, of designing custom imaging sensors for robots. The integration itself will prove the concept of the system. The Retina is the basic building block for a real-time interactive observer.

7 Conclusions and future plans

The development of an Active Observer is underway at the GRASP laboratory. Although future emphasis will be placed on the control structure of such an observer, its integration policies, and communication issues with other observers and agents in general, there is still a need for further studies, developments and improvements of component technologies. For example, in the case of understanding surface reflectance, we still have not completed the theoretical underpinning of transparency. With the problem of segmentation, while the cooperation between surface and volumetric fittings is necessary, and they help in resolving ambiguities, the first and second order primitives are clearly not sufficient for modeling a broad class of real life objects. Higher order models will have to be invoked, but only selectively and locally after the lower order fits have failed. If this order of fitting data is violated then instabilities in the fitting procedures can be expected. Finally, there is the question of the control mechanism of the Active Observer. As shown above, we have employed the Discrete Event Dynamic System model. DEDS is a suitable formalism to model continuous processes of observation, as well as events occuring in discrete intervals. As a result, this model allows us to predict the observation capability as defined by the control theory community. The assumption here, however, is that the task of observation is *a priori* in terms of the discrete events. While in the original theory the transitions from one state/event to another were discrete, we have extended the theory to transitions with uncertainties. The next task should be to loosen the requirements for explicit knowledge of the desired observable events. These events should be able to be generated from some rules of physics, geometry and other conventions of the object's and agent's interactions. In conclusion, we are on our way to complete an Active Observer which has a control structure that allows us to predict observation capabilities. The components developed here allow the Active Observer to handle moderately complex scenes of shapes/materials, their spatial arrangements and their illuminations. The real time issue of processing is a crucial one and hence our efforts in special purpose CCD chips and related hardware. The open questions are many but we wish to concentrate on the intercommunication of several observers and other agents, such as manipulatory, mobile and human agents. Ultimately, the final issue is this: who tells what and how much, and to whom.

References

[1] J. Aloimonos and A. Badyopadhyay. Active vision. In *Proc. 1st Int. Conf. on Computer Vision*, pages 35–54, 1987.

[2] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):996–1005, 1988.

[3] R. Bajcsy, S.W. Lee, and A. Leonardis. Color image segmentation with detection of highlights and local illumination induced by inter-reflections. In *Proc. 10th International Conf. on Pattern Recognition*, Atlantic City, NJ, June 1990.

[4] E. N. Coleman and R. Jain. Obtaining 3-dimensional shape of textured and specular surface using four-source photometry. *Computer Graphics and Image Processing*, 18(4):308–328, 1982.

[5] R. Gershon. *The Use of Color in Computational Vision*. PhD thesis, Department of Computer Science, University of Toronto, 1987.

[6] G.H. Healey and T.O. Binford. Using color for geometry-insensitive segmentation. *Journal of the Optical Society of America*, 6, 1989.

[7] T. Kanade and K. Ikeuchi. Introduction to the special issue on physical modeling in computer vision. *IEEE Trans. PAMI*, 13(7):609–610, 1991.

[8] G.J. Klinker, S.A. Shafer, and T. Kanade. Image segmentation and reflection analysis through color. In *Proceedings of the DARPA Image Understanding Workshop*, pages 838–853, Pittsburgh, PA, 1988.

[9] S. K. Nayar, K. Ikeuchi, and T. Kanade. Determining shape and reflectance of hybrid surfaces by photometric sampling. *IEEE Trans. Robo. Autom.*, 6(4):418–431, 1990.

[10] S.A. Shafer. Using color to separate reflection components. *COLOR Research and Application*, 10(4):210–218, 1985.

[11] H.D. Tagare and R. J. deFigueiredo. Photometric stereo for diffuse non-lambertian surface. *IEEE Trans. PAMI*, 13():, 1991.

[12] L. B. Wolff. *Polarization Methods in Computer Vision*. PhD thesis, Department of Computer Science, Columbia University, 1991.

[13] Gupta, Alok, *Surface and Volumetric Segmentation of Complex 3-D Objects Using Parametric Shape Models*, Technical Report MS-CIS-91-45, Department of Computer and Information Science, University of Pennsylvania, 1991.

[14] Gupta, Alok and Ruzena Bajcsy, Part description and segmentation using contour, surface and volumetric primitives, in *Proceedings of the Conference on Sensing and Reconstruction of 3D Objects and Scenes*, pp. 203–214, SPIE, Santa Clara, CA, Feb 1990.

[15] Gupta, Alok, Luca Bogoni, and Ruzena Bajcsy, Quantitative and qualitative measures for the evaluation of the superquadric models, in *Proceedings of the IEEE Workshop on Interpretation of 3D Scenes*, pp. 162–169, Austin, TX, November 1989.

[16] Leonardis, Ales, Alok Gupta, and Ruzena Bajcsy, Segmentation as the search for the best description of the image in terms of primitives, in *Proceedings of the Third International Conference on Computer Vision*, pp. 121–125, IEEE, Osaka, Japan, December 1990a.

[17] Leonardis, Ales, Alok Gupta, and Ruzena Bajcsy, *Segmentation of Range Images as the Search for the Best Description of the Scene in Terms of Geometric Primitives*, Technical Report MS-CIS-90-30, CIS Department, University of Pennsylvania, 1990b.

[18] Solina, Franc, *Shape Recovery and Segmentation with Deformable Part Models*, PhD thesis, University of Pennsylvania, 1987, Technical Report MS-CIS-87-111.

[19] Solina, F. and R. Bajcsy, Recovery of parametric models from range images: the case for superquadrics with global deformations, *IEEE Trans. on Pattern Analysis and Machine Intelligence, 12*, 131–147, February 1990.

[20] R. Bajcsy and T. M. Sobh, *A Framework for Observing a Manipulation Process*. Technical Report MS-CIS-90-34 and GRASP Lab. TR 216, Computer Science Dept., School of Engineering and Applied Science, University of Pennsylvania, June 1990.

[21] T. M. Sobh, *A Framework for Visual Observation*. Technical Report MS-CIS-91-36 and GRASP Lab. TR 261, Computer Science Dept., School of Engineering and Applied Science, University of Pennsylvania, May 1991.

[22] T. M. Sobh and R. Bajcsy, "Visual Observation of A Moving Agent". In *Proceedings of the European Robotics and Intelligent Systems Conference (EURISCON '91)*, Corfu, Greece, June 1991 and presented at the 12^{th} *International Joint Conference on Artificial Intelligence (IJCAI)*, Workshop on Dynamic Scene Understanding, Sydney, Australia, August 1991.

[23] T. M. Sobh and R. Bajcsy, "A Model for Observing a Moving Agent". In *Proceedings of the Fourth International Workshop on Intelligent Robots and Systems (IROS '91)*, Osaka, Japan, November 1991.

[24] J. Van der Spiegel, G. Kreider, et al. "A Foveated Retina-Like Sensor Using CCD Technology". In *Analog VLSI Implementations of Neural Systems*, ed. C. Mead and M. Ismail, pp. 189-211, Kluwer Academic Publishers, Boston, 1989.

[25] G. Kreider, J. Van der Spiegel et al. "The Design and Characterization of a Space Variant CCD Sensor". SPIE Vol. 1381 *Intelligent Robots and Computer Vision IX: Algorithms and Techniques*, Boston, November 1990.

[26] G. Kreider, J. Van der Spiegel et al. "A Retina-Like Space Variant CCd Sensor". SPIE Vol. 1242 *Charge Coupled Devices and Solid State Optical Sensors*, pp. 133-140, Santa Clara, February 1990.

[27] Z. Kalayjian. "A Driver Circuit for the Foveated Retina-Like Optical Sensor". Final Report, Undergraduate Fellowship in Sensor Technologies, University of Pennsylvania, 1990.

Vision-Based Navigation*

William B. Thompson
Department of Computer Science
University of Utah
Salt Lake City, UT 84112

Herbert L. Pick, Jr.
Center for Research in Learning
Perception, and Cognition
University of Minnesota
Minneapolis, MN 55455

Abstract

Our research focuses on navigation in outdoor terrain. We are particularly concerned with viewpoint determination in the absence of easily and uniquely identifiable landmarks. Progress has been made on methods for extracting topographic features, on the development of effective strategies for accurate viewpoint determination, and in characterizing localization errors. These results are relevant to problems in mobile robotics, navigation aids, and training.

1 Introduction

Vision-based navigation is a difficult and challenging problem, particularly in large-scale outdoor domains. The Universities of Utah and Minnesota are jointly conducting research on navigation in outdoor terrain. The current focus is on *localization* tasks in which a viewpoint must be determined given a map of the area and imagery available at the viewpoint [Heinrichs *et al.*, 1989].

Two general approaches to localization are possible. *Signal-based* methods (e.g., [Ernst and Flinchbaugh, 1989, Andreas *et al.*, 1978, Baird and Abramson, 1984]) correlate expected imagery (or other data) generated from terrain models with that actually coming from sensors. Signal-based approaches work best when there are good *a priori* estimates of current location, reducing the combinatorics associated with correlation over viewing position and direction. Even then, it can be difficult to generate a sufficiently accurate expected view to compare against actual data using any efficiently computed matching metric. Signal-based methods have been most successful in applications such as cruise missile guidance where active sensing is used to gather range "images" which can also easily be synthesized from elevation maps. *Feature-based* methods first extract salient features from the map and the available imagery and then match these features rather than the raw data itself. They are particularly appropriate when definable landmarks are present and where the available sensing modalities are such that photometrically accurate synthetic views cannot be produced. Feature-based approaches to object recognition have recently been receiving substantial interest (e.g., [Grimson, 1990]), since a careful choice of features can significantly reduce sensitivity to image properties not relevant to the task at hand.

Our work is principally directed at feature-based methods for localization. Three sub-problems need to be solved [Thompson, 1990, Thompson *et al.*, 1990]. Salient features relevant to localization must be extracted from imagery. This is in principal a relatively standard low-level vision task, though the image understanding community has done only limited work to date on extracting such features from ground-level images. Equivalent features need to be extracted from digital elevation models or other available maps. The cartographic community has done related work, but not specifically in support of localization. Finally, correspondences must be established between map and image features and these correspondences then used to determine the viewpoint. The closest related work to this is in photogrammetry (e.g., [Sanso, 1973, Thompson, 1958]) and alignment approaches to object recognition (e.g., [Huttenlocher and Ullman, 1987, Grimson, 1990]). Neither of these sources, however, provides much help in developing methods for establishing feature correspondences in terrain data while avoiding prohibitive combinatorial difficulties.

Results obtained over the last year include:

- Feature selection and grouping strategies have been developed which reduce the combinatorics of feature matching while at the same time simplifying viewpoint determinations [Smith *et al.*, 1991, Heinrichs *et al.*, 1992, Pick *et al.*, 1992].
- We have shown that the choice of the particular landmarks used for localization can have a dramatic effect on the precision with which localization can be accomplished [Sutherland, 1992, Sutherland and Thompson, 1992].
- A method has been developed for extracting relevant features from elevation data which combines local and global effects to maximize saliency. Preliminary results have been obtained in extracting image features which are specifically relevant to the localization task in outdoor terrain [Savitt *et al.*, 1992].

*This work was supported by National Science Foundation grant IRI-9196146, with partial funding from the Defense Advanced Research Projects Agency, and by a grant from Texas Instruments.

2 Strategies for Localization

Landmark-based localization involves a combinatorial matching problem in which features from a map and features from views must be put into correspondence. For localization to succeed, these correspondences must be correct and must yield an accurate estimate of the viewpoint. In the absence of unique cultural features, solving the correspondence problem is difficult because individually identifiable topographic features are not typically present in views of open terrain. For example, while a number of hills may be visible from a particular viewpoint, the views seldom provide enough information about the detailed shape of each hill to find the single corresponding pattern in a map.

Two important consequences follow from this analysis:

- Correspondences must be established in a manner that minimizes ambiguity.
- Since ambiguity cannot be eliminated, it must be possible to generate and then evaluate multiple hypotheses about the location of the viewpoint.

These objectives can be achieved by exploiting the power of a number of simple matching strategies. Many of these strategies will seem obvious, but few have previously been explicitly incorporated into models of localization. Matching should be initiated by starting with features in the view and using these to drive a search for the corresponding map features. This is because features in the view are far more likely to be relevant than features on the map, since most map features will not be visible from any single viewpoint. Information about the immediate surroundings, often obtainable with reasonable accuracy, can significantly constrain possible viewpoints. Such constraints can be incorporated into the path planning process to generate movement which facilitates localization. The manner in which multiple hypotheses are compared is critical. In particular, it is most important to note expectations which are *not* met. Finally, correspondence is aided when individual features are assembled into configurations of multiple features. These configurations should have both viewpoint invariant properties that help in searching over the map and viewpoint dependent properties that constrain possible viewpoints.

3 Sensitivity in Viewpoint Determination

Assuming that correct correspondences between visible landmarks and map features have been established, the viewpoint must still be determined. In principle, this involves straightforward trigonometric computations. However, effective localization also requires an understanding of the errors that can occur. Often, distance estimates to landmarks will not be available. Measurements of absolute bearing to a single landmark or the visual angle between landmarks will always have some amount of uncertainty associated with them. This uncertainty propagates through the localization computation and translates into uncertainty about the actual viewpoint.

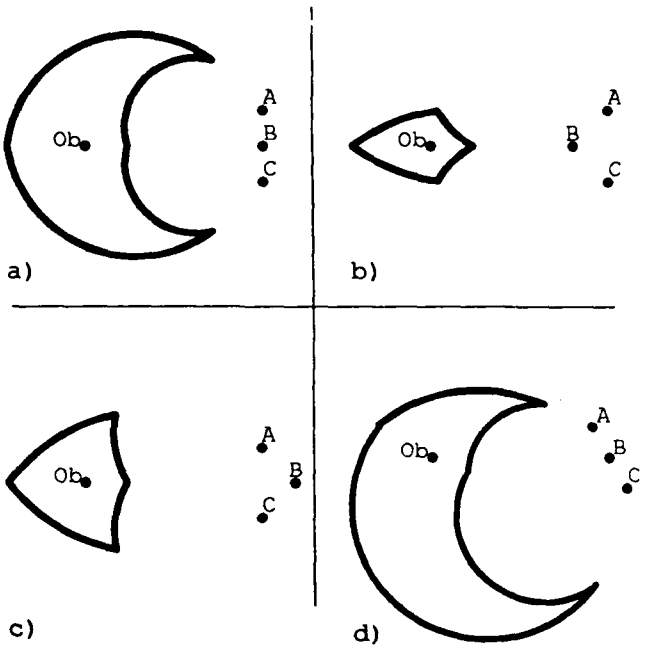


Figure 1: Regions of viewpoint uncertainty as a function of landmark configurations.

We have completed an error analysis of localization based on relative bearings to landmarks all within a field of view $< 180°$. The nature of the error was found to vary substantially with the particular configuration of landmarks used. Figure 1 provides four examples. In each case, Ob marks the actual observation point. A, B, and C indicate the location of three landmarks. In all four cases, the observation point is approximately the same distance from the landmarks and the landmarks are approximately the same distance from each other. The dark line circumscribes the region of uncertainty associated with an error of $\pm 30\%$ in the estimation of the visual angle between landmarks. That is, any location within the dark lines is consistent with visual angles that vary no more than 30% from the actual angles observed at Ob.[1]

Figure 1a shows the situation when observations are made to the side of an equally spaced, linear configuration of landmarks. Substantial localization uncertainty results. Figures 1b and 1c are similar, except that landmarks are no longer arrayed along a straight line. Figure 1d shows another linear configuration of landmarks, this time viewed from a position offset to the side. Not only is the region larger than in Figure 1a, but the area of uncertainty is skewed away from the "true" position.

When there is a choice of landmarks available on which to base a localization determination, it is important to choose the configurations that lead to the least uncertainty. Furthermore, the general nature of the uncertainty regions can be predicted based on simple proper-

[1] An error bound of $\pm 30\%$ in visual angle is realistic when dealing with non-point features such as rounded hills. Even when the bearing to landmarks can be better localized, the same qualitative differences in uncertainty regions hold.



Figure 2: Ridgelines and valleys using local slope properties.



Figure 3: Ridgelines and valleys using hydrologic simulation.

ties of the configuration of landmarks.

4 Extraction of Map and Image Features

Vision-based navigation requires image understanding techniques to extract landmark features from views of the environment and map understanding methods to locate corresponding features in cartographic data. While existing segmentation algorithms can be modified to deal with the first of these tasks, new approaches are required for map understanding.

The cartographic community has developed a number of methods for efficiently encoding elevation data. Less attention has been paid to the extraction of landmarks specifically useful for navigation. We know that peaks, ridgelines, saddle points, and valleys are important in localization. Each of these has a well specified definition in terms of differential operators applied to the elevation surface [Haralick *et al.*, 1983]. Implementations of these operators work fairly well for detecting point features such as peaks and saddle points in digital elevation models. They are less effective in extracting line features such as ridgelines and valleys. The reason for this is that the operators are computed over a local, symmetric neighborhood. The saliency of ridge and valley topography to navigation, however, is a function of both local and global effects. For example, the importance of a ridgeline has to do with the sharpness of the ridge, its height above the surrounding terrain, and its length. A local differential operator can only measure the sharpness.

We have developed a novel method of extracting valley features that employs a "reverse engineering" heuristic based on the processes that created the valley in the first place. A hydrologic simulation is done to determine the flow that would result if the local topography was

covered with a uniform density of fluid.[2] We have found that high flow rates in the simulation are associated with valleys that are of a magnitude sufficient to make them priority candidates as landmarks. Ridgelines are found using a variation of this method that is less closely related to the terrain forming geological processes but is nevertheless effective in finding salient features. In effect, the fluid flow is now "up hill", with appropriate modifications to deal with the fact that ridgelines do not vary monotonically in elevation along their length.

Figure 2 shows valleys (coded in white) and ridges (coded in black), extracted from a digital elevation model using local differential detectors. Figure 3 shows the corresponding features obtained with the hydrologic flow method. (Not shown is the fact that this method gives a measure of importance to each point on the ridge and valley features.) The features in Figure 3 are connected, as they should be, and as shown in [Savitt *et al.*, 1992] better correspond to significant topographic structures.

5 Relevance

The work outlined above has relevance to a wide variety of applications. When large numbers of potential landmarks are available to a navigation system, our work on error analysis and sensitivity provides criteria for selecting those landmarks which will yield the highest expected precision. Such information is useful in both mission planning situations and for automated navigation aids. The extraction of salient topographic features from digital elevation data may be of use in both navigation aids and training. The problem solving strategies which make possible efficient feature-based solutions to navigation problems will be important to future automated systems which use such a feature-based approach. More immediately, it may be possible to use an understanding of these strategies, determined from a computational analysis of the problem, to improve education in map usage. Finally, this research contributes to basic science in image understanding. Localization provides a well-defined task with clear criteria for success enabling the effective evaluation of the associated image understanding techniques. In addition, a formal analysis of vision-based navigation problems reveals similarities with other computer vision tasks. As an example, our work on error and sensitivity has relevance to pose estimation in object recognition.

References

[Andreas *et al.*, 1978] R. D. Andreas, L. D. Hostetler, and R. C. Beckmann. Continuous Kalman updating of an inertial navigation system using terrain measurements. In *Proc. IEEE National Aerospace Electronics Conference*, pages 1263–1270, 1978.

[Baird and Abramson, 1984] C. A. Baird and M. R. Abramson. A comparison of several digital map-aided navigation techniques. In *Proc. IEEE Position Location and Navigation Symposium*, pages 294–300, 1984.

[Ernst and Flinchbaugh, 1989] M. D. Ernst and B. E. Flinchbaugh. Image/map correspondence using curve matching. In *AAAI Symposium on Robot Navigation*, pages 15–18, March 1989.

[Grimson, 1990] W. E. L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints.* The MIT Press, 1990.

[Haralick *et al.*, 1983] R.M Haralick, L.T. Watson, and T.J. Laffey. The topographic primal sketch. *Int. Journal of Robotics Research*, 2(1), Spring 1983.

[Heinrichs *et al.*, 1989] M. R. Heinrichs, D. R. Montello, C. M. Nusslé, and K. Smith. Localization with topographic maps. In *Proceedings of the AAAI Symposium on Robot Navigation*, pages 29–32, March 1989.

[Heinrichs *et al.*, 1992] M. R. Heinrichs, K. Smith, H. L. Pick, Jr., and B. H. Bennett. Strategies for solving localization problems. In this proceedings, 1992.

[Huttenlocher and Ullman, 1987] D. P. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proc. First International Conference on Computer Vision*, pages 102–111, 1987.

[Pick *et al.*, 1992] H. L. Pick, Jr., M. R. Heinrichs, D. R. Montello, K. Smith, C. N. Sullivan, and W. B. Thompson. Reading topographic maps. In J. Flach, P. Hancock, J. Caird, and K. Vicente, editors, *Ecology of Human-Machine Systems.* 1992.

[Sanso, 1973] F. Sanso. An exact solution of the roto-translation problem. *Photogrammetria*, 29:203–216, 1973.

[Savitt *et al.*, 1992] S. L. Savitt, T. C. Henderson, and T. L. Colvin. Feature extraction for localization. In this proceedings, 1992.

[Smith *et al.*, 1991] K. Smith, M. R. Heinrichs, and H. L. Pick, Jr. Similarity judgment and expert localization. In *Proceedings Thirteenth Annual Conference of the Cognitive Science Society*, August 1991.

[Sutherland and Thompson, 1992] K. T. Sutherland and W. B. Thompson. Sensitivity of feature configuration in viewpoint determination. Submitted to the IEEE Conference on Computer Vision and Pattern Recognition, June, 1992.

[Sutherland, 1992] K. T. Sutherland. Sensitivity of feature configuration in viewpoint determination. In this proceedings, 1992.

[Thompson *et al.*, 1990] W. B. Thompson, H. L. Pick, Jr., B. H. Bennett, M. R. Heinrichs, S. L. Savitt, and K. Smith. Map-based localization: The "drop-off" problem. In *Proc. DARPA Image Understanding Workshop*, pages 706–719, September 1990.

[Thompson, 1958] E. H. Thompson. An exact linear solution of the problem of absolute orientation. *Photogrammetria*, 13(4):163–178, 1958.

[Thompson, 1990] W. B. Thompson. Vision-based navigation. In *Proc. DARPA Image Understanding Workshop*, pages 141–143, September 1990.

[2] The simulation algorithm has a time complexity that is linear with respect to the size of the digital elevation map. As a result, it can be implemented quite efficiently.

Research at Brown University*

David B. Cooper, Thomas L. Dean, William A. Wolovich

Brown University,
Providence, RI 02912

Abstract

We continue to explore the issues involved in building integrated planning, sensing, and control systems for applications in service robotics. The project is applicable to materials handling robotics, ground or aerial information gathering robotics, and other. The robotics applications serve as a practical focus and impetus for combining individual efforts in image understanding, tactile sensing, planning, and control. One common effort is focussed on a small mobile platform mounting a pair of stereo cameras, a low resolution laser rangefinder, and a mechanical force sensor. The goal here is model-based navigation through corridors and rooms, and recognition and position estimation of objects to be acquired. The algorithms and systems being developed for this purpose are specialized versions of more general results coming from a number of research directions. Those reported on in this overview are the following.

1. Vision: Major progress was made in: 2D curve and 3D surface object modeling, robust low computational cost recognition, and analysis based on implicit polynomials of degree higher than 2; robust low computational cost estimation of optical flow for motion detection and object tracking; highly complex curve shape modeling and recognition based on reaction-diffusion equations.
2. Mechanical Sensing: 3D objects can now be recognized by measuring planar surface curves at different elevations with a continuous contact probe, and inferring the surface from the curves.
3. Planning and System Integration: We have developed efficient decision-theoretic methods for generating multistep plans that involve information gathering and the control of sensing. These methods are being applied to mobile robotics problems involving conflicting tasks and requiring relatively short response times.
   A low computational cost algorithm finding paths through a region populated by curved complex shaped objects has been developed by modeling objects as implicit polynomial surfaces.

Other work in new areas in vision and in planning and system integration is in progress, and will be presented in the next reporting period.

1 Vision

1.1 Implicit Polynomial Curves of Degree Higher Than Two

Implicit higher degree polynomials in x, y, z (or in x, y for curves in images) have considerable global and semiglobal representation power for objects in 3D space. (Spheres, cylinders, cones and planes are special cases of such polynomials restricted to second degree.) Hence, they have great potential for object recognition and position estimation. In [13, 14, 15, 16], Taubin presented a beautifully organized easily accessible introduction to these polynomials and some of their properties, and developed low computational cost algorithms that give excellent curve fits to data in the plane and non-planar data in 3D, and surface fits to range data in 3D. Though the polynomial curve and surface fits to the data are stable in the vicinity of the data and fit the data remarkably well, in practice we find that the coefficients of these higher degree polynomials may be sensitive to small changes in the data. This poses a problem since we would like to compare curves and surfaces based on the ***polynomial coefficients only***. Since the last IU Workshop, we have worked on and solved three problems to overcome this difficulty and enable the use of these polynomials in real world robust systems [4, 12]: 1) Characterization and fitting algorithms for the subset of these algebraic curves and surfaces that is bounded and exists largely in the vicinity of the data; 2) A Mahalanobis distance for comparing the coefficients of two polynomials to reliably determine whether the curves or surfaces that they represent are close over a specified region; 3) Geometric Invariants for determining whether one implicit polynomial curve or surface is

*This work was supported in part by NSF-DARPA Grant #IRI-8905436, NSF Grant #IRI-8715774, NSF Presidential Young Investigator Award #IRI-8957601, DARPA Grant #F30602-91-C-0041, and by NSF-DARPA Grant #IRI-8905436

a rotation and translation of another, or whether one implicit polynomial curve is an affine transformation of another. Unlike the invariants from classical invariance theory, ours are functions of all the coefficients for a polynomial of arbitrary degree, and require modest computation. In addition to handling objects with easily detectable features such as vertices, high curvature points, and straight lines, the polynomials and tools discussed in this paper are ideally suited to smooth curves and smooth curved surfaces which do not have detectable features. The motivation for this work was for a system that could handle very general object shapes. This work is presented in this workshop proceedings in [5].

1.2 Practical Optical Flow for Motion Detection and Tracking

Many current optical flow algorithms are not suited for practical implementations such as tracking because they either require massively parallel supercomputers, specialized hardware, or up to several hours on a scientific workstation. One particular reason for this is the quadratic nature of the search algorithms used in these problems.

Recent work has developed two modifications to these types of algorithms which can convert quadratic-time optical flow algorithms into (at worst) linear-time ones. The first uses a variable image sampling rate which trades space for time and yields an algorithm that is at worst linear, and at best constant, in the speed of the moving objects in the image. This technique finds the fastest motion in an image and is ideal for tracking, since the fastest moving objects in a robot's environment are generally the most interesting.

The second modification extends this approach to create a multiple-speed optical flow field by transforming two-dimensional searches over space into one-dimensional searches in time. This space-time inversion has the effect of searching for faster moving objects in each image before searching for slower moving ones, with additional effort being exerted to search for slower objects only when desired. A system of velocity masking allows a tradeoff of angular resolution (but not magnitude resolution) in the optical flow field for an algorithm that is only linear, rather than quadratic, in the range of velocities present.

These algorithms yield real-time performance on standard serial hardware, and are suitable as input to higher level routines such as tracking modules and depth determination based on relative motion.

1.3 Exploring the Shape Manifold: the Role of Conservation Laws

People have the remarkable ability to recognize objects from their two-dimensional projections. This is often despite the absence of other visual cues like texture, color, stereo, motion, etc. Although shape plays a significant role in recognition, viable theories for representing and describing shape have been elusive. We believe that the key to robust object recognition is to explicitly capture the inherent connectivity that exists among similar shapes. For example, shapes generated as an object is being occluded, stretched, bent, or dented are all similar; this similarity relationship should be explicitly represented in the *space of shapes*. We propose an abstract mathematical framework for shape which leads to the computation of a robust representation suitable for object recognition. Our approach relies on axioms motivated by properties of objects and their projections. The relationship between an object and its neighbors in this abstract space is described by slight deformations. As such, we study deformations of objects and as a first step, those that change a shape's boundary solely as a function of its local geometry.

We have shown that arbitrary deformations of curves as a function of their local geometry is to a first degree captured by combinations of deformations along the normal that are either constant or depend on curvature [6]. It is in the interaction between constant and curvature deformations that such issues as boundary/region, local/global, and process/parts are resolved. Our study of the evolution of curves under such deformations and their local behavior [8] has led to an intriguing result. Namely, this evolution satisfies a viscous conservation law which depicts a two dimensional space, our *reaction-diffusion space*. Since these deformations are nonlinear they lead to singularities. A notion of *entropy* for shape is developed which limits the singularities of shape to *shocks*. The formation of shocks and their classification in the reaction-diffusion space is the basis of our representation for shape. As a special case the locus of shocks gives the medial axis but which in addition is "colored" with *type* and *significance*. As another application, a notion of scale for approximating shape naturally emerges from our framework (based on entropy) leading to a two-dimensional *entropy scale space*. Interestingly, certain morphological operations as well as Gaussian smoothing methods are special cases in this space [7]. It is in this space that the idea of a shape *similarity metric* becomes viable. Our computational results also suggest a language for describing shape in which *parts*, *protrusions*, and *bends* are the basic elements, and the perceptual reality of which is illustrated via qualitative experiments. Finally, from a numerical perspective, our simulations are robust under low resolution (128x128) and high noise, and are potentially suitable for applications involving object recognition and classification.

2 Continuous-Contact Tactile (Force/Torque) Sensing

The current focus of this research has been directed at expanding the two dimensional algorithm described in last year's report to enable a robot to track and to recognize three dimensional objects. Initially, the objects considered are spheres, cylinders, cubes, cones and variations of these.

The three dimensional object tracking algorithm is meant to be as general as possible. It does not depend on an object's orientation, shape or location in the workspace; i.e. there is no prior knowledge of the object's shape or surface contour. In our initial experiments, all objects are tracked using the two dimensional tracking algorithm. The robot moves along an object's surface in a plane parallel to the earth's surface, and planar slices are taken at increasing levels in the vertical

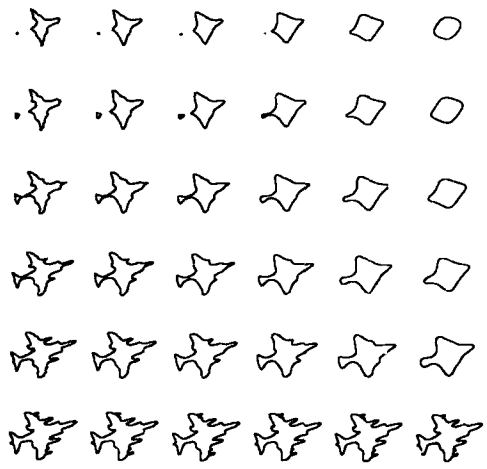


Figure 1: This figure illustrates a portion of the reaction diffusion space for the shape of a fighter jet. The horizontal axis is a measure of reaction (left) versus diffusion (right) and the vertical axis is time (amount of deformation). Note how shocks form and disappear as one traverses this space.

direction. The location of the object and the change in shape of each planar slice is identified from the information gathered during the tracking. Information about the change in the surface contour in the vertical (or any) direction can be extrapolated from explicit knowledge of the Cartesian position of each planar slice. Given this information, the recognition program is able to distinguish between simple objects, such as spheres, cylinders, cones, and boxes.

Simple three dimensional objects have been used to test the tracking algorithm. For example, planar slices from a box were identified as quadrilaterals and the data points from a cylinder were recognized as circles. Hence, simple objects can be recognized by combining shape information from the planar data slices.

Complex objects also have been employed in the experiments. A standard telephone receiver has been placed at different orientations in the workspace and surface contour data has been collected using a modified dual-drive tracking control algorithm. This data will be used in an implicit polynomial model-based object recognizer. This work is detailed in [10, 11]

3 Planning, System Integration and Navigation

3.1 Navigating a Robot between Curved Irregular Obstacles

In this work an attempt is made to plan smooth collision-free paths for mobile robots between obstacles described by polynomials. Although the idea of using algebraic techniques to describe obstacles is not new, previous work is theoretical in nature and does not reflect on how to describe arbitrary, non-polygonal obstacles. Also, previous algebraic-based methods take at least exponential time to find a collision free path. This work overcomes these problems. it does so by utilizing two powerful mathematical tools to solve the problem of navigating a mobile robot between irregulary shaped obstacles. The first is *description of obstacles by polynomials*, which enables a very compact description of fairly complicated obstacles, as well as a rule for very quickly deciding whether the robot touches an obstacle or not. The second tool is *regularization* to choose a smooth solution from a multitude of possible solutions. In our case regularization will assist in choosing a short and smooth path among the paths constituting a solution, e.g. connecting the starting and goal positions and avoiding the obstacles. It is known that the shortest path problem is NP-complete; regularization promises a reasonably short path. A marked advantage of the work presented here is that it allows to describe these seemingly different requirements from a path - its smoothness and non-intersection with obstacles - *in the same language*, which is a simple algebraic expression in the path points, thus enabling a unified treatment of these two requirements. The algorithm has a running time that is linear in the number of obstacles.

3.2 Planning and System Integration

We have adopted Bayesian decision theory as the basis for communication between modules performing various sensing and interpretation tasks and those engaged in decision making. Modules performing sensing and interpretation are prompted for information concerning spatial or object properties and return hypotheses and associated distributional information (e.g., an object of type T is located in region R with probability better than π).

For instance, in the process of locating a particular sought-after object, a decision making module might ask a module responsible for interpreting range data from stereo for information about the location of three planar patches in a particular spatial configuration. The stereo module would return a set of hypotheses, where each hypothesis consists of a 3-D description for the three planar patches, and a measure of confidence for each hypothesis. The decision making module would use the information returned by the stereo module to confirm (or disconfirm) one of its own hypotheses; it might commit to a particular hypothesis (e.g., reporting on the location of the sought-after object) or it might engage in further information gathering (e.g., moving the robot to obtain a different view and then asking some module—not necessarily the stereo module—for additional information [12]). If, for example, low-error object identification is required, the decision module might use several stereo image pairs taken from different positions to generate a small number of candidate positions for the sought after object, and then employ a more time consuming test involving tactile recognition to eliminate spurious candidates. The interpretation of range data from stereo constitutes a relatively quick-and-dirty estimate, while the tactile sensing provides high-accuracy information even in low- or no-light situations.

In our framework, decision modules routinely perform value-of-information calculations to decide whether to perform additional information gathering tasks. In all of the applications we are looking at, the robot is interacting with other processes outside of its control, and so it is constantly pressed for time. Information costs

in terms of time and opportunities lost while obtaining that information.

In [2], we describe our first mobile robotics application involving a robot given the task of tracking and periodically reporting on the position of a mobile target. Performance is measured in terms of the accuracy of the reports. The robot is given a representation (map) of its environment, but has to estimate its location in that environment by continually observing its surroundings and matching against the information in the map. Since its primary sensor for obtaining observations of its environment is an array of sonar sensors, the robot can easily get lost when moving in large open areas. The robot tracks the target using multiscale optical flow calculations [1].

At each point in time, the robot has to make tradeoffs that involve losing registration with the map (getting lost) and losing contact with the target. In both cases, losing registration and losing contact, the accuracy of the robot's reports on the location of the target suffer. Determining an optimal or near-optimal course of action requires careful consideration of the robot's surroundings and perceptual capabilities. By employing a general graph-based representation for Bayesian decision models, we are able to carefully analyze the complexity of the decision models required. The decision model that we provide for this problem is quite general and, at least in theory, provides an optimal decision procedure. In practice, we were forced to make concessions to the combinatorics inherent in the problem. In [9], we describe those concessions in some detail, justifying the tradeoffs that we made in terms of the potential loss of precision versus the potential reduction in decision-making response time.

The tracking problem described above was designed as a (relatively) simple exercise involving decision making and active perception in time-stressed situations. With the exception of the techniques for multiscale optical flow, the task does not involve any image understanding. We are continually reviewing our image understanding and tactile sensing research in an effort to find and transfer techniques to our service robotics efforts. In particular, we are seeking to develop interpretation and parameter estimation routines that run on the computational resources available in our mobile robotics test environment. We are developing fast, robust procedures for estimating the parameters of surface patches from stereo pairs, tracing surface contours using tactile sensing, and extracting depth information from optical flow fields.

We have recently begun work on a new application involving a mobile robot engaged in reconnaissance operations. In this proceedings [3], we investigate some of the representational issues that arise in the reconnaissance task as well as more general issues that arise in quantifying (gathering statistics for) Bayesian decision models and designing efficient inference algorithms for problems in robotics.

References

[1] T. Camus. Applications of pyramid structures to multiscale optical flow. Technical Report CS-90-09, Brown University Department of Computer Science, 1990.

[2] T. Dean, T. Camus, and J. Kirman. Sequential decision making for active perception. In *Proceedgs of the DARPA Image Understanding Workshop*, pages 889-894., 1990.

[3] T. Dean and J. Kirman. Representation issues in bayesian decision theory for planning and active perception. In *Proceedings of the DARPA Image Understanding Workshop*. DARPA, 1992.

[4] D. Keren, D. Cooper, and J. Subrahmonia. Describing complicated objects by implicit polynomials. Technical Report LEMS-93, Brown University, 1991.

[5] D. Keren, J. Subrahmonia, G. Taubin, D. B. Cooper. Bounded and unbounded implicit polynomial curves and surfaces, mahalanobis distances, and geometric invariants, for robust object recognition. This Proceedings.

[6] B. B. Kimia, A. Tannebaum, and S. W. Zucker. Toward a computational theory of shape: An overview. In *Proceedings of the First European Conference on Computer Vision*, Antibes, France, 1990.

[7] B. B. Kimia, A. Tannebaum, and S. W. Zucker. Entropy scale-space. In *Proceedings of the Intl. Workshop on Visual Shape*, Capri, Italy, May 1991. Plenum Press.

[8] B. B. Kimia, A. Tannenbaum, and S. W. Zucker. On the evolution of curves via a function of curvature, I: The classical case. *JMAA*, To appear, 1991.

[9] J. Kirman, K. Basye, and T. Dean. Sensor abstractions for control of navigation. In *Proceedings of the IEEE International Conf. on Robotics and Automation*, 1991.

[10] K. Korzeniowski and W. Wolovich. 3-D object recognition using dual-drive control. Submitted for presentation at the 1992 IFAC Symposium on Intelligent Components and Instruments for Control Applications.

[11] K. Korzeniowski and W. Wolovich. 3-D surface tracking using dual-drive control. In *Proceedgs of the 30th IEEE Conf. on Decision and Control*, Dec 1991.

[12] J. Subrahmonia, D. Cooper, and D. Keren. Reliable object recognition using high dimensional implicit polynomials for 2D curves and 3D surfaces. Technical Report LEMS-94, Brown University, 1991.

[13] G. Taubin. 3D object recognition and positioning with algebraic invariants and covariants. IBM Report RC-16211. Presented at NSF Workshop on The Integration of Numerical and Symbolic Computing Methods, Saratoga Springs, July 1990. To appear in *Symbolic and Numerical Computations-Towards Integration*, Academic Press.

[14] G. Taubin. *Recognition and Positioning of Rigid Objects using Algebraic and Moment Invariants.* PhD thesis, Brown University, 1990.

[15] G. Taubin. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:1115—1138, November 1991.

[16] G. Taubin and D. B. Cooper. 2D and 3D object recognition and positioning system based on moment invariants. In *Proceedgs. DARPA-ESPIRIT Workshop on Geometric Invariants, Rikjavik*, March 1991. To appear in *Geometric Invariance in Machine Vision*, MIT Press.

[17] Cooper, D., Cernuschi-Frias, B., Hung., Y. P., Subrahmonia, J. Use of Markov Random Fields in Estimating and Recognizing Objects in 3D Space, In *Markov Random Fields : Theory and Application*, Edited by R. Chellapa and A. Jain, Academic Press, 1991.

SECTION II

Current IU Applications

The OSD Tactical Unmanned Ground Vehicle Program

LTC Erik G. Mettala, Ph.D.
Deputy Director
Software and Intelligent Systems Technology Office
Defense Advanced Research Projects Agency

Abstract

The Department of Defense Unmanned Ground Vehicle Program is a multi-year effort involving the Services, OSD and DARPA. The objective of the program is to first field a teleoperated unmanned ground vehicle system, followed by preplanned product improvements leading to self-navigating systems performing reconnaissance, surveillance, target acquisition and designation missions. The system capabilities envisaged are enabled, in large part, by image understanding, planning, control and robotics research performed over the last ten years. The program will continue to push the state of research, while significantly advancing the state of practice.

1.0 Background

In recent years, Congress "has been concerned about the direction and composition of the many diverse robotics projects undertaken by the armed services and defense agencies."[1] Congress therefore directed the establishment of a DoD robotics master plan in 1989. The diversity of robotics projects that were described in the 1989 plan led to Congressional request in 1990 to consolidate all of the ground robotics vehicle projects "under OSD policy and program direction."

In response to this Congressional mandate, previously separate ground vehicle related robotics efforts were consolidated in a single program element, under the direction of the Tactical Warfare Programs (TWP) office of the Director for Defense Research and Engineering (DDR&E). Since 1990, the TWP office has been responsible for reporting on the activities of this program element to Congress, providing direction, allocating appropriated funds to projects, and carefully monitoring the progress of all DoD Unmanned Ground Vehicle activities. The Services and the Defense Advanced Research Projects Agency (DARPA) are responsible for the conduct and daily management of the projects.

1.1 Recent Planning

Following the consolidation of DoD robotics activities, a number of planning activities took place involving OSD, the Services and DARPA. The overall rationale for the robotics program was developed, based on analysis of existing and emerging requirements, coupled with an assessment of emerging image understanding, control, and planning technologies. The various projects underway or proposed by the Services were reviewed. Several on-going programs were terminated, and those that were selected for continuance were restructured to produce a "a more focused and cost-effective robotics program," as directed by Congress.

Prior to the consolidation of robotics efforts, all robotics and unmanned ground vehicle programs in DoD were principally technology driven, as opposed to operating under the mandate of user requirements. Since consolidation, the U.S. Army has approved a requirement for an unmanned ground vehicle, named "CALEB," to conduct reconnaissance, surveillance and target acquisition (RSTA) in support of ground infantry forces.

National economic pressures, coupled with world-wide events leading to the dismantling of the Warsaw Pact, and events leading to the secession of virtually all republics from the USSR are forcing the DoD to reduce overall force structure. However, as the events of Desert Shield/Storm have shown, the United States still must maintain the

1. Report 101-132 from the Senate Committee on Appropriations on the Department of Defense Appropriations Bill, 1990. Quotations is this section are from Report 101-132.

capability to move large, technologically sophisticated forces to counter the actions of hostile nations.

1.2 Impact of Desert Storm

Many lessons concerning the relative importance and likelihood of future robotic systems were demonstrated in Desert Storm. Among them:

- For the first time, Unmanned Air Vehicles (UAVs) were widely used in combat.
- Land forces confronted the immediate threat of chemical weapons.
- A hurried request for remotely operated mine-clearing tanks, and shallow watercraft was made.
- High technology weapons demonstrated the effectiveness of autonomous guidance.
- Desert Storm revealed the political significance of individual weapons systems in regional conflicts, with the SCUD/PATRIOT events being the most dramatic example.
- Desert Storm set a standard of minimal friendly casualties against which the results of future conflicts will be measured.

1.3 UGV/TUGV Program Structure

The technology associated with unmanned systems[1] is maturing faster than the concepts of employment are being developed. The structure of the DoD Unmanned Ground Vehicle Program reflects this reality, in a coordinated evaluation and development program with the objective of first fielding of an unmanned system by 1998.

The TUGV is the principal effort of the current UGV advanced development program. The TUGV program is being planned and managed with an awareness that it represents an initial step in the evolution and fielding of UGVs for combat applications and that its success or failure may have far-reaching consequences.

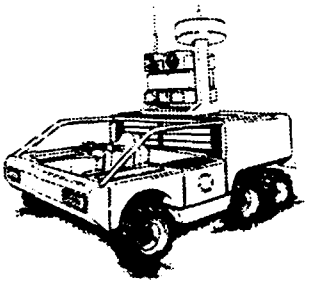
Three principal foci encompass the DoD UGV program. First, several Surrogate Teleoperated Vehicles (STVs) will be developed and used to support Early User Test and Evaluation (EUTE) of UGV concepts. Second, a full scale Engineering and Manufacturing Development (EMD) program will develop the first fielded Teleoperated Unmanned Ground Vehicle. Third, the Unmanned Ground Vehicles Technology Enhancement and Exploitation (UGVTEE) program will focus on maturing those robotics technologies of particular interest to UGV systems. The UGVTEE program is a demonstration-directed effort, including Demo-I and Demo-II, whose principal aims are to mature and transition near-term technology, and to develop semi-autonomous navigation technology respectively.

1. The term UGV is used in a general sense to include a range of applications. The term Tactical Unmanned Ground Vehicle (TUGV) will refer to a specific project, which is developing one class of UGVs.

2.0 The Surrogate Teleoperated Vehicle (STV) Program

The STV program, managed by the Joint Unmanned Ground Vehicles Office (JUGVO) at the U.S. Army Missile Command (MICOM) in Huntsville, Alabama, will develop 14 Surrogate Teleoperated Vehicles (STVs). These will be used to conduct Early User Test and Evaluation (EUTE), by placing six STVs in a USMC infantry brigade, and six in a U.S. Army brigade for a period of one year, starting in 1992.

FIGURE 1. The Surrogate Teleoperated Vehicle



As shown in Figure 1, the STV is a six-wheel-drive, fully amphibious platform. It contains all automotive and navigational components, including sensors and control for teleoperated driving under day, night, and adverse environmental conditions. The platform is powered by a hybrid 25 horsepower (HP) diesel engine and a 3 hp electric motor, for silent locomotion when required. Automotively, the STV will be able to traverse roads at 35 miles per hour (mph) and travel off-road at 25 mph. Its remote driving speeds will depend: 1) on the skill of the operator in teleoperated mode or, 2) on the sophistication of the software as semi-autonomous capabilities are added. By placing these teleoperated vehicles into the hands of soldiers and marines, the JUGVO will acquire direct access to employment concepts created by users in tactical environments.

3.0 TUGV Engineering and Manufacturing Development

In the Engineering and Manufacturing phase, the selected system contractor(s) will be responsible for fabricating a production-ready TUGV. The Government will conduct Developmental Test and Evaluation and Initial Operational Test and Evaluation of the contractor-provided TUGV prototypes. These tests will determine readiness for production of a first generation TUGV. Milestone III is planned for the end of 1997.

4.0 UGV Technology Enhancement and Exploitation

UGVTEE consists of technology base efforts supporting current and future UGV projects, and involve participation from academe, industry, DoD, DoE and NASA laboratories. The UGVTEE program is directed to exploit robotics advances and mature those technologies that are critical to the robotization of UGV systems. The near term focus of this program is on providing the mission capabilities and technological enhancement required for the TUGV. This part of the program (Demo-I) will conclude in FY1992. The long term focus is on image understanding, planning and control technologies that will enhance operational capability and survivability. This part of the program (DEMO-II) has been initiated in FY1991, with the main focus of developing autonomous navigation under battlefield conditions. Technology development will include support for reconnaisance, surveillance, and acquisition (RSTA) functions while the UGV is moving, and distributed artificial intelligence supporting automated communication with other vehicles, and work load partitioning between vehicles to accomplish mission objectives.

5.0 UGVTEE Demo-I

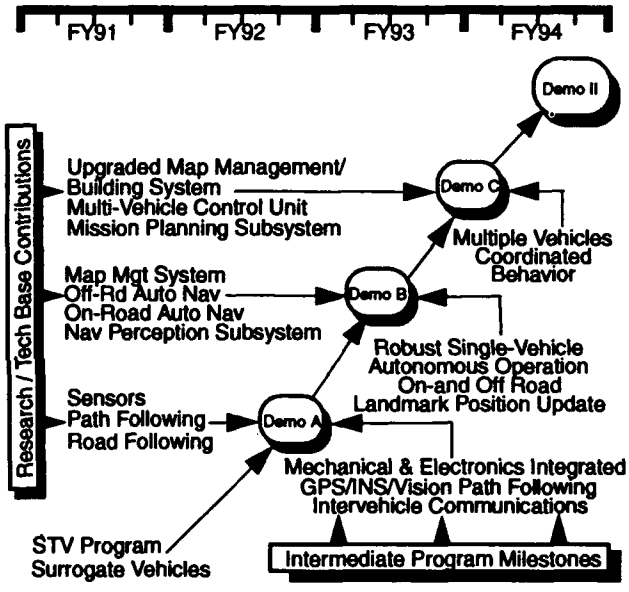
The principal purpose of Demo-I is to mature critical system component technologies for first generation teleoperated UGVs and demonstrate their readiness for acquisition programs. Based on the results of Demo-I, selected technologies will be integrated into the basic STV for the development of a complete TUGV prototype. The emphasis is on reducing operator work load while enhancing performance of the RSTA mission.

6.0 UGVTEE Demo-II

The purpose of Demo-II is to develop and mature those navigation technologies that are critical to evolving UGVs from labor intensive teleoperated systems requiring fibre-optic cables for communication to supervised autonomous systems utilizing low-bandwidth non-line-of-sight communication. The objective of the program will be to demonstrate four semi-autonomous cooperating unmanned ground vehicles performing navigation, reconnaissance, surveillance, target acquisition and target designation.

As shown in Figure 2, Demo-II is a four-phase five year program with three interim demonstrations directed at transitioning research results onto the surrogate vehicles. In each of the sections describing the interim demonstrations that follow, the objectives and technological approach for each will be discussed.

FIGURE 2. Demo-II Overview



6.1 Demo-II Technologies

Realization of the Demo-II objectives will require moderate to substantial increases in capabilities from current state-of-the-art in Image Understanding, Navigation, Planning, Control, and Distributed Artificial Intelligence. The technological basis upon which the recommendations for approving the Demo-II program were based on research results developed under support by the DARPA Image Understanding[1], Planning[2], and Robotics Science[3] Programs, and others.[4]

Of principal importance in the consolidation of the DoD program, has been the research results demonstrated on the Carnegie Mellon University Navlab, shown in Figure 3. Navlab is a mobile research laboratory, designed for facilitating research in navigation, control, and image understanding.

FIGURE 3. CMU Navlab



6.1.1 Stereo Vision.

Emerging requirements for all military systems require that principles of stealth, low observability, and low emission be observed. Consequently, a significant objective of the Demo-II program is to develop passive stereo vision based techniques for acquiring the local environment models required by unmanned ground vehicles for navigation and obstacle avoidance. While much progress has been made in stereo vision, differences in technical approach and hardware requirements demand that the current state of research be stressed, that metrics be established, and that the alternative approaches be quantifiably measured against these metrics. The Demo-II program is specifically funding research in real-time stereo vision and stereo analysis to overcome the existing limitations in analysis of stereo vision.

---

1. Principal Image Understanding Research provided by Carnegie Mellon University [Kanade, Thorpe, Whittaker], University of Massachusetts [Hanson, Riseman, Weems], University of Maryland [Rosenfeld, Davis], University of Pennsylvania [Bacjsy], University of Rochester [Ballard, Brown], Advanced Decision Systems [Morgan], General Electric Corporation [Mundy], SRI [Bolles, Hannah, Strat], Hughes Aerospace Company [Tseng, Nash], Massachusetts Institute of Technology [Poggio, Lozano-Perez], and others.

2. Principal Planning Research provided by University of Maryland [Davis], Carnegie Mellon University [Thorpe, Whittaker], Massachusetts Institute of Technology [Brooks], Advanced Decision Systems [Soldo], University of Michigan [Jain, Durfee], Yale University [Dean], Stanford University [Cannon, LaTombe], and others.

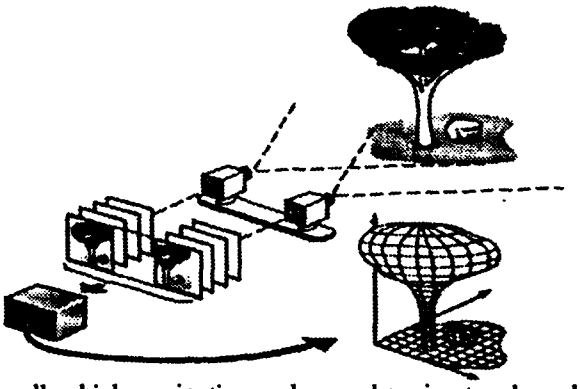
3. Principal Robotics Science Research provided by University of Pennsylvania [Bacjsy], Stanford University [Cannon], Massachusetts Institute of Technology [Brooks, Lozano-Perez], and others.

4. My sincerest apologies to those not listed due to minimal space.

Current research indicates that it is now becoming feasible to use range images obtained with passive stereo vision to detect obstacles in support of semi-autonomous, cross-country navigation. Within the Demo-II program, we are focusing on methods to characterize the performance of stereo vision quantitatively, and means to develop design methodologies that relate task requirements to stereo system design parameters.

Our objective, as shown in Figure 4, is to develop an engineering approach to designing passive stereo vision systems that provide sufficient detail to enable successful

FIGURE 4. Stereo Vision



small vehicle navigation under overhanging tree branches, and to detect obstacles such as rocks of a size that is relevant to the current navigation task - far enough ahead of a vehicle to ensure safety at computationally and visually supportable speeds. Such an engineering approach includes calculating the vehicle stopping distance as a function of initial velocity, calculating the minimum look-ahead distance at which sensors must detect obstacles, and to determine detectability criteria as a function of look-ahead distance, obstacle size and sensor noise characteristics. Finally, the approach leads to specification of sensor system parameters, such as image resolution and computational speed that is required to meet the detectability criteria for the desired vehicle speed.

Multiple technical approaches are currently under investigation [Hannah 89, Matthies 91, Nishihara 90]. Each emerging approach will be evaluated against the engineering criteria as discussed, and further characterized as to actual performance in a system configuration that is supported by the overall Demo-II effort.

6.1.2 Image Understanding

The four vehicles participating in Demo-II will be equipped with two sensor packages, one for navigation (NAV), and one to support the vehicle primary mission of reconnaissance, surveillance and target acquisition, (RSTA). Image Understanding plays an important, if not

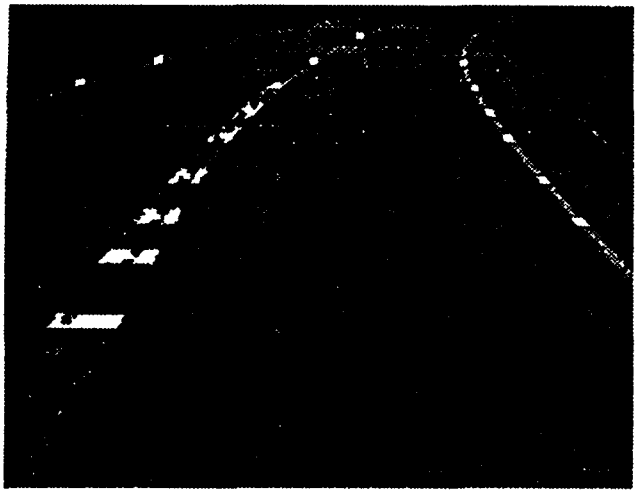
dominant role in each of these primary systems. IU provides indicators in the NAV system, aiding the determination of the class of environment in which the vehicle is operating, and providing identification of road signs, vehicles, and terrain features supporting planners for rules of the road, obstacle interaction, and vision based position updates respectively.

Each vehicle will be expected to navigate in on-road and off-road environments utilizing the NAVsubsystem, while simultaneously surveying areas of interest for threat vehicles or targets with the RSTA subsystem. Having determined the existence of a target, the RSTA subsystem will be designed to facilitate target segmentation, tracking while on the move, and designation with a laser target designator. Additionally, target classification and identification will be provided, to the extent technologically feasible, to indicate to the operator a suggestion of the type of target confronting the vehicle.

6.1.3 Navigation

Navigation research, coupled closely with results from the image understanding community, has formed the technical basis for the investment in the Demo-II efforts. For example, Figure 5 illustrates work on YARF, or Yet Another Road Follower [Kluge 90]. YARF explicitly models as many aspects as possible for driving on structured roads. YARF has individual knowledge sources that know how to model and track specific features, such as road edge markings (white stripes); road center lines (yellow stripes); and shoulders. YARF also uses an explicit geometry model of the road, consisting of location of vehicle on road; location of stripes; type of stripes (e.g., broken or solid); and maximum and current road curvature.

FIGURE 5. Yet Another Road Follower (YARF)



6.1.4 Ladars

The driving speed of the vehicle will be limited by the obstacle detection capability, the reliability, and the speed of the range sensors. Laser radar (LADAR) is an active range sensor that can be used for both obstacle detection and landmark identification [Besl 88, Hebert 88]. In general, LADAR technology seems to be relatively mature and reliable. Since LADAR employes active illumination, performance at night should be as good as during the day, if not better (since the sun is a noise source during the day). The data density, the percentage of pixels for which a range estimate is available should be quite high for moderate scenes (approximately 90%) . Signal to noise considerations cause the standard deviation of range measurements to be approximately a quadratic function of the true range.

The field of view (FOV) of the ERIM laser scanner used in the ALV program is 80 degrees horizontally and 30 degrees vertically, with a spatial resolution of 256 pixels horizontally (about 0.3 degrees/pixel) and 64 pixels vertically (about 0.5 degrees/pixel). The ERIM scanner has a maximum range of 20 meters to the first ambiguity interval and a range resolution quantized to about 8 centimeters. It produced the image in 0.5 seconds.

A scanner made by Odetics provides 128 x 128 pixels in a 60 x 60 degree field of view, giving an angular resolution of about 0.5 degrees. It has an ambiguity interval of 9.4 meters and a quantization interval of 1.8 centimeters..

6.1.5 Mission Level Planning

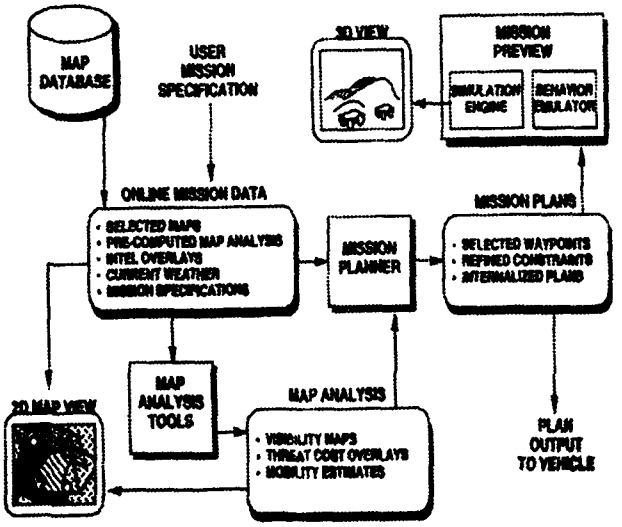
Mission planning utilizes digital terrain data, DTED level II, augmented with Interim Tactical Data (ITD) containing soils, roads, draining, foliage and water characteristics. The planningsystem, illustrated in Figure 6, has the following components:

- Mission Specification
- Mission Analysis
- Mission Planning
- Mission Execution
- Plan Evaluation & Revision

Mission specification is supported by 2-D graphics input procedures, augmented with 3-D views of selected maps, pre-computed map analyses, intelligence data, and current weather. Mission analysis is performed by map analysis tools, which generate visibility maps, threat cost overlays and mobility estimates for the mission area. Mission planning itself utilizes online mission data and the products of map analysis to generate selected waypoints, refined constraints and internal plan specifications for each of the SSVs participating in the mission.

A unique feature of the planning system is the mission preview that is provided to the operator. Similar in concept to SIMNET, the operator may view any portion of the spatio-temporal projection of the mission in a 3-D view, and is provided with a viewport facility that enables an aerial perspective of the area of operations during mission preview, as well as during mission execution. The preview facility also provides for after-action reporting and mission review, following capture of vehicle position data, target nomination, and map annotation identification reports by the vehicles during mission execution.

FIGURE 6. Mission Planning System



During mission execution, each of the vehicles interacts with the OCU, providing position updates and video feedback to the operator on demand. Status data for each of the vehicles is provided, including window-oriented textual, 2-D graphical position description and RSTA module viewpoint, and updated indicators marking targets and objects of interest in the 3-D view.

The plan evaluation and revision system utilizes the 'flying carpet' view to display the mission to the operator. During simulated mission execution, the operator is provided with the utilities to refine mission parameters, revise constraints, and update the individual vehicle plans.

6.1.6 Enroute Planning and Control

Once the UGV begins to traverse the planned path, autonomous navigation functions depend on sensor inputs and algorithms for processing these inputs to ascertain and characterize the local environment. This includes local three-dimensional map generation from stereo vision and/or LADAR images; model-based environment characterization with object and scene identification and tracking; landmark recognition; and global and local map merging to upgrade the terrain map and determine vehicle location and orientation. Local path planning uses the terrain map, the sensed environment, vehicle position, and planning constraints. *Driving algorithms include:*

1. Neural network and feature-based road following, allowing the vehicle to travel on paths ranging from multilane paved highways to gravel roads to jeep trails, traverse intersections, and recognize road branches.

2. Road and cross-country navigation based on "fuzzy routes or spline driving" instructions with position provided by GPS receivers, inertial measurement systems, odometry, and vision-activated landmark position estimates.

3. Vision-guided obstacle avoidance

4. Execution of local path plan timed to the global mission plan and appropriately adjusted to terrain, visibility, path surface condition, automotive and dynamic constraints, and limitations on concurrent processing of the large amount of data required for the vehicle navigation and RSTA function.

6.1.7 Processing Technology

DoD has an ongoing research and development program to develop a succession of prototypes of scalable parallel and distrivbuted hetrogeneous high performance computing systems and associated software and algorithms for military applications. These systems are developed with progressively larger scale, more advanced components, more dense packaging, and more advanced architecture. DEMO II will use the most current prototype that is suited for TUGV applications. Under consideration are the Intel iWarp and an Image Understanding Architecture (IUA) processor developed jointly by the University of Massachusetts and Hughes.

As shown in Figure 7, the Image Understanding Architecture (IUA) consists of three levels of tightly coupled parallel processors (for low level, intermediate level, and high level vision operations [Weems 89]). The IUA proof-of-concept prototype has been completed and experience with both the hardware and extensive software simulations are guiding the development of a second generation of the IUA that will be used in the UGV program. Furthermore, the initial research-oriented software development environment is currently being replaced by a sophisticated set of applica-

tion-oriented tools. Thus, the IUA effort is in the process of making the transition from an isolated research project to being in a position of accessibility to the wider community. In this proceedings, Weems et al [Weems 92] describe the current status of the effort and some of the plans for the future. The second generation IUA retains the basic three-level structure of the prototype, but the new system will be computationally more powerful than the prototype and will be much easier to use for applications. The IUA development is taking place at three sites: the University of Massachusetts at Amherst, Hughes Research Laboratories in Malibu, California, and Amerinex Artificial Intelligence Inc at Amherst.

The coterie network for the iWarp is shown in Fig. 8.

FIGURE 7. Image Understanding Architecture

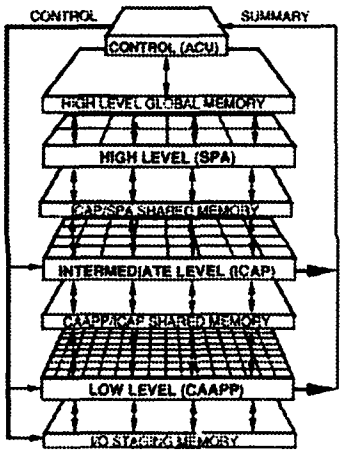
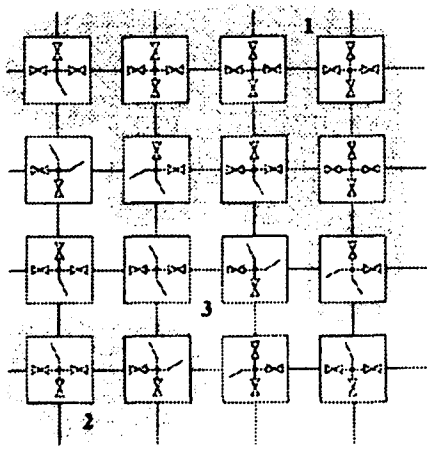
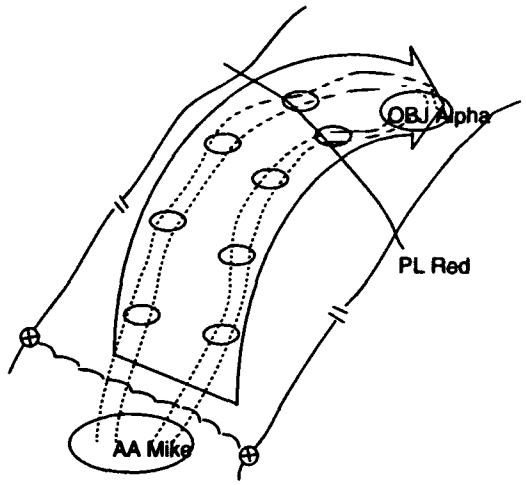


FIGURE 8. Coterie Network



6.2 The Demo-II Program

FIGURE 9. Mission Context



The Demo-II program will provide the operator a man-portable operator control unit, with capabilities for displaying a simulated view of the area of interest of the battlefield, overlaid on request with direct video feedback from one or more RSTA or NAV system sensors carried by the SSVs. Figure 9 illustrates the mission planning context that might be presented to the operator, providing normal military symbology for boundaries, assembly areas, objectives, phase lines, and so forth. The detailed terrain model underlying the planning system will be acquired by UAV overflight of the area of interest, and transmittal of overhead imagery to the OCU for further processing to develop a properly annotated terrain model. The dashed lines in Figure9 illustrate the individual vehicle paths that are computed using "Stealth Terrain Navigation" procedures, such as those described in these proceedings by Davis [Davis 92].

The operator interacts with the OCU to provide the specification of the plan, verifies acceptability of the path plans from the planner, and commands the OCU to download the individual vehicle plans onto each of the vehicles participating in the current mission.

6.2.1 Offensive Operation

Demo-II will operationally demonstrate a movement to contact by a team of four cooperating UGVs. The team will conduct a screening operation for a manned force using bounding overwatch, or similar military tactics. Semi-autonomous navigation will over semi-arid terrain traversing a distance of some 10 KM at an average speed, while maneuvering, of 40 KMH. Once in overwatch positions, the team will employ the RSTA mission module

to observe threats; locate, detect, assess, and designate for engagement by indirect fire.

FIGURE 10. Navlab-II



6.2.2 Defensive Operation

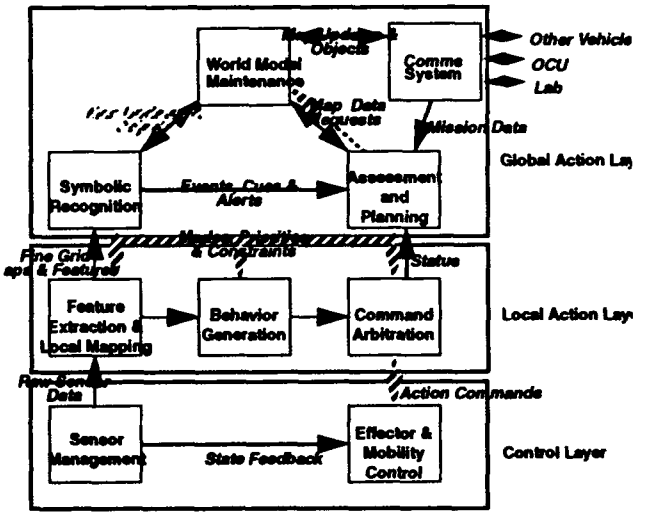
Demo-II will also operationally demonstrate a retrograde operation by a team of four cooperating UGV's. The team will screen a manned force by sequentially occupying preplanned defensive positions, or order, to maximize degradation of enemy forces. Once the UGV-supported commander determines that enemy forces have been attrited, the team will move to pre-planned locations in the main battle area to designate and defeat remnants of advancing enemy forces.

6.2.3 Architectural Overview

Figure13 represents the SSV Top-Level Architecture. This diagram, and substantial accompanying material, was developed by the SSV Architectural Tiger Team[1] in response to a DARPA mandate to construct a high-level view of the system architecture. The goal of this effort was to capture the essence of the technical elements of the program, provide a means of comprehensible explanation, and resolve the seemingly diminishing, yet ever-present conflicts between layered and behaviorist approaches to robotics architectural specification.

The resulting architectural specification provides a clear framework for describing the technological approach of the program. Based on a three-tiered hierarchy, the essence of the specification is to include behavioral generation and command arbitration elements in the central layer. The separation into layers is not intended to be a subsystem partitioning, but rather, represents a way of classifying subsystems and managing their interfaces. Elements of the layers (boxes in the diagram) represent subsystems. Elements of the subsystems (boxes underlying the boxes in Figure11) are objects.

FIGURE 11. SSV Top-Level Architecture



Layer View

The *control layer* contains all of the system's "autonomic" functions. The highest rate elements of the system reside here -- nominally all servo loop closures and compensation. Processes in this layer tend to be running in synchronous real time, facilitating control observability. The implementation will likely be multi-rate, since different control systems have differing closure loop rate requirements. Low-level safety and consistency checks are also included in this layer.

The *local action layer* of the system contains mainly behavioral and reflexive elements. Most data in this layer (i.e. local world model) is ephemeral. The "world model" at this level is, for the most part, implicit in the intermediate representation of this layer. For instance, grid-based maps are very finely resolved (perhaps 20 centimeters) and continuously rolled over as the vehicles traverses to prevent data overload. Functions in this layer operate at a lower rate than the control layer, but nominally still in synchronous real time (with provision for event- or signal-driven functions). This layer may also be multimodal, responding to mode commands or constraints from the Global Action Layer.

The concept of a "behavior" is loosely defined at this level. By "behavior," we refer to the generation of some command or sequence which may ultimately be committed to action, depending on the priorities and modes of the system's current state. These behaviors vary from the very

1. Bob Bolles, SRI; Larry Matthies, JPL; Rich Luhrs, MMC; Chuck Thorpe, CMU; Monet Soldo, ADS; David Payton, HRL

currently being replaced by a sophisticated set of application-oriented tools. Thus, the IUA effort is in the process of making the transition from an isolated research project to being in a position of accessibility to the wider community. In this proceedings, Weems et al [Weems 92] describe the current status of the effort and some of the plans for the future. The second generation IUA retains the basic three-level structure of the prototype, but the new system will be comp_tationally more powerful than the prototype and will be much easier to use for applications.

The coterie network of the IUA, shown in Fig. 8, is part of the IUA that provides network support for region-based operations. For example, on the IUS's lowest level, a program can segment an image into regions, and then issue one instruction to compute the size of a region which is simultaneously executed in all regions. The Coterie Network explicitly connects pixels in each region (and conversely inserts breaks between regions). .

The IUA development is taking place at three sites: the University of Massachusetts at Amherst, Hughes Research Laboratories in Malibu, California, and Amerinex Artificial Intelligence Inc at Amherst.

FIGURE 7. Image Understanding Architecture

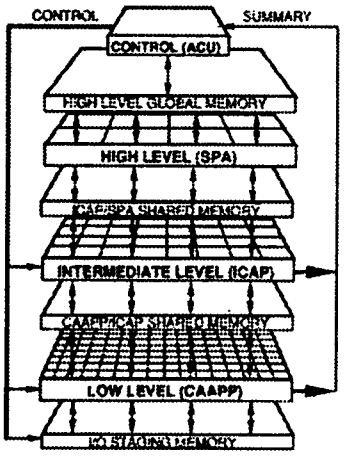
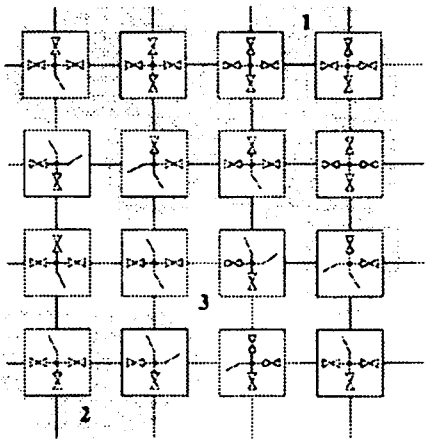
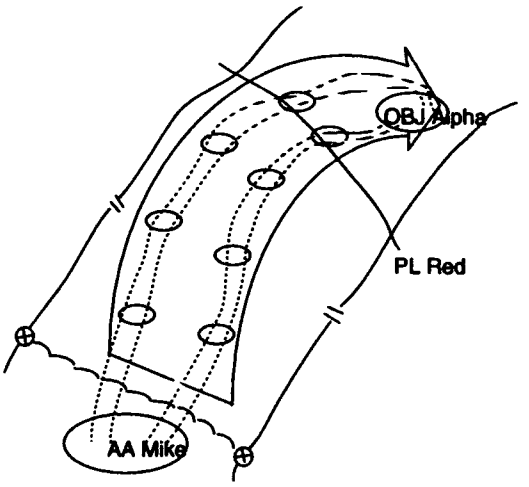


FIGURE 8. Coterie Network



6.2 The Demo-II Program

FIGURE 9. Mission Context



The Demo-II program will provide the operator a man-portable operator control unit, with capabilities for displaying a simulated view of the area of interest of the battlefield, overlaid on request with direct video feedback from one or more RSTA or NAV system sensors carried by the SSVs. Figure 9 illustrates the mission planning context that might be presented to the operator, providing normal military symbology for boundaries, assembly areas, objectives, phase lines, and so forth. The detailed terrain model underlying the planning system will be acquired by UAV overflight of the area of interest, and transmittal of overhead imagery to the OCU for further processing to develop a properly annotated terrain model. The dashed lines in Figure9 illustrate the individual vehicle paths that

are computed using "Stealth Terrain Navigation" procedures, such as those described in these proceedings by Davis [Davis 92].

The operator interacts with the OCU to provide the specification of the plan, verifies acceptability of the path plans from the planner, and commands the OCU to download the individual vehicle plans onto each of the vehicles participating in the current mission.

6.2.1 Offensive Operation

Demo-II will operationally demonstrate a movement to contact by a team of four cooperating UGVs. The team will conduct a screening operation for a manned force using bounding overwatch, or similar military tactics. Semi-autonomous navigation will over semi-arid terrain traversing a distance of some 10 KM at an average speed, while maneuvering, of 40 KMH. Once in overwatch positions, the team will employ the RSTA mission module to observe threats; locate, detect, assess, and designate for engagement by indirect fire.

FIGURE 10. Navlab-II



6.2.2 Defensive Operation

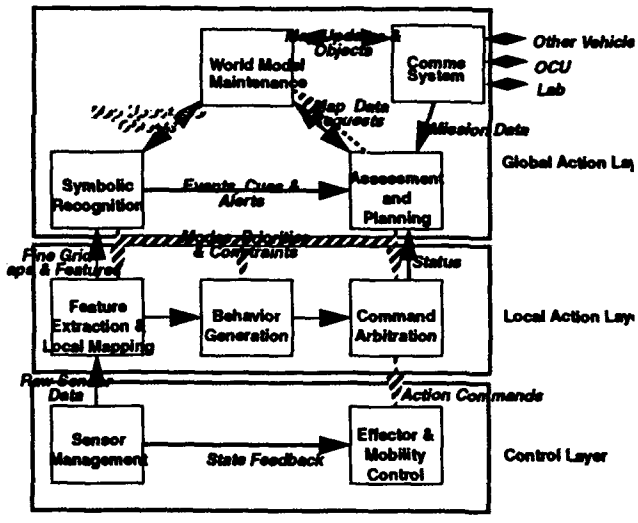
Demo-II will also operationally demonstrate a retrograde operation by a team of four cooperating UGV's. The team will screen a manned force by sequentially occupying preplanned defensive positions, or order, to maximize degradation of enemy forces. Once the UGV-supported commander determines that enemy forces have been attrited, the team will move to pre-planned locations in the main battle area to designate and defeat remnants of advancing enemy forces.

6.2.3 Architectural Overview

Figure13 represents the SSV Top-Level Architecture. This diagram, and substantial accompanying material, was developed by the SSV Architectural Tiger Team[1] in response to a DARPA mandate to construct a high-level view of the system architecture. The goal of this effort was to capture the essence of the technical elements of the program, provide a means of comprehensible explanation, and resolve the seemingly diminishing, yet ever-present conflicts between layered and behaviorist approaches to robotics architectural specification.

The resulting architectural specification provides a clear framework for describing the technological approach of the program. Based on a three-tiered hierarchy, the essence of the specification is to include behavioral generation and command arbitration elements in the central layer. The separation into layers is not intended to be a subsystem partitioning, but rather, represents a way of classifying subsystems and managing their interfaces. Elements of the layers (boxes in the diagram) represent subsystems. Elements of the subsystems (boxes underlying the boxes in Figure11) are objects.

FIGURE 11. SSV Top-Level Architecture



Layer View

The *control layer* contains all of the system's "autonomic" functions. The highest rate elements of the system reside here -- nominally all servo loop closures and compensation. Processes in this layer tend to be running in synchronous real time, facilitating control observability. The implementation will likely be multi-rate, since different control systems have differing closure loop rate requirements. Low-level safety and consistency checks are also included in this layer.

1. Bob Bolles, SRI; Larry Matthies, JPL; Rich Luhrs, MMC; Chuck Thorpe, CMU; Monet Soldo, ADS; David Payton, HRL

The *local action layer* of the system contains mainly behavioral and reflexive elements. Most data in this layer (i.e. local world model) is ephemeral. The "world model" at this level is, for the most part, implicit in the intermediate representation of this layer. For instance, grid-based maps are very finely resolved (perhaps 20 centimeters) and continuously rolled over as the vehicles traverses to prevent data overload. Functions in this layer operate at a lower rate than the control layer, but nominally still in synchronous real time (with provision for event- or signal-driven functions). This layer may also be multimodal, responding to mode commands or constraints from the Global Action Layer.

The concept of a "behavior" is loosely defined at this level. By "behavior," we refer to the generation of some command or sequence which may ultimately be committed to action, depending on the priorities and modes of the system's current state. These behaviors vary from the very simple (such as emergency stop for sudden appearance of an obstacle directly in the vehicle's path) to fairly complex (such as formation following with other vehicles). The different levels of complexity will in all likelihood require multi-rate operation when implemented in real-time processes.

Though at first glance this layer appears to imply a strictly behaviorist approach, provision is made for a purely projective planning approach through the implementation of a "Plan Execution" behavior. The "behavior" becomes dominant, though some low-level reflexive behaviors, such as emergency stop, may still be active for vehicle safety reasons. In this mode, the planning function in the Global Action Layer develops highly integrated motion and activity plans, which are executed by Plan Execution as long as the "Situation Assessment" function is satisfied that current plans can be satisfied. Lower-level planning functions (such as trajectory planning) may be implemented in the Local Action Layer due to information-sharing or timing requirements. Also in this mode, "Command Arbitration" is simplified, as there is little to arbitrate other than the reflexive behaviors.

The *global action layer* contains the "cognitive" or "reasoning" elements of the system. It incorporates object recognition, ATR functions, landmark recognition and pose refinement, and other "sensor fusion" tasks. Projective planning elements (route planning, path planning, and activity planning) are found in this layer, as well as the persistent data (the explicitly represented world model) and any "reasoning" about that model, such as required for multivehicle coordination, situation assessment, mission and plan monitoring. These functions are the lowest rate elements of the system, and tend to be event-driven, aperiodic processes.

Module View

The *sensor managment* module is responsible for managing and directing the data intensive sensory input from the navigation and RSTA mission modules. The navigation sensor package is equipped with a stereo pair color camera, a stereo pair image intensifier, a ladar, accelerometers and position encoders to report current position and velocity measurements.

The *feature extraction and local mapping* module encompases a very broad range of utilities, including image preporcessing, ortho-rectification, local map matching and building, first alert to significant events, terrain classification, annotated map maintenance, image stabilization, subimage selection, image warping, stereo range extraction, motion analysis, road edge detection, and also serves as the imput layer for combined neuraly network/rule based approaches to navigating in structured domains.

The *behavior generation* module supports cross-country trajectory evaluation, road tracking, object tracing, stereo system control, emergency action generation, and low-level planning for obstacle avoidance.

The *command arbitration* module supports plan execution, and arbitration modules for speed, steering, sensor pointing and access.

The *symbolic recognition* module supports landmark recognition, object recognition, map data abstraction, target acquisition, target tracking, and target recognition.

The *assessment and planning* module encompasses sensor management, mission evaluation, behavior prioritization, route and path planning, activity scheduling, situation assessment, plan monitoring, and multivehicle coordination.

World model maintenance includes global map updating functions, global position maintenance, object list maintenance and tracking, and also supports map database query handling.

The *effector and mobility controls* include functional control of pan/tilt actuation loop closures, discrete controls such as sensor on/off, driving controls, vehicle plant controls, lens and camera controls, vergence controls for the stereo systems supporting driving and RSTA, manipulator controls, and safety features and authority limits supporting robust safety considerations for the vehicles.

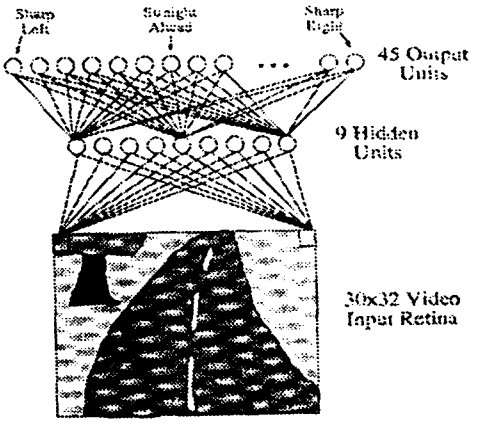
6.2.4 Phase I and Demo A - Basic Navigation, Positioning and Path Following

Phase I of the Demo-II program will culminate in Demo A, to be conducted in September 1992. This effort will integrate basic mechanical and electrical components, automotive controls, basic color and infrared visual sensors, GPS, INS, odometry, and intervehicle communications required to support the foundation of autonomous navigation. Many known research results will be ported onto the STVs, including autonomous inertial path following, and autonomous visual path following based on spectral, model, and neural net approaches. After Demo A, the vehicles built for Demo-II will be known as Surrogate Semi-Autonomous Vehicles (SSVs)[1].

Demo-A will provide for independent verification of local action and control layer navigation "behaviors." The initial behavior to be verified incorporates the basic mobility functions of accurate positioning and path following. These follow from research conducted by Stentz and Whittaker at Carnegie Mellon University known as "FASTNAV." FASTNAV is based on Kalman state space estimation from inertial measurement units, GPS receivers, and odometry. The path following behavior computes error terms from desired spatio-temporal positions with respect to the path plan loaded on the vehicle by the mission planning system. These error terms are used in conjunction with combined feedback and feed-forward controllers to implement a path-tracker.

A second navigation behavior to be evaluated is based on research in Autonomous Land Vehicle in a Neural Network, or ALVINN [Pommerleau 90, Pommerleau 92], see Figure 12. The color reflectance values found in the video frame buffer are filtered in the blue spectral region, then a weighted average of pixels is computed, mapping the input video buffer to a 30 x 32 video input retina in ALVINN. Each of the 960 nodes in the input retina is fully connected to nine hidden units, each of which is in turn fully connected to all of the output units. The output layer is a linear representation of the currently appropriate steering direction. The centermost output unit represents the "travel straight ahead" condition, while units to the left and right of the center represent successively sharper left and right turns. The steering direction dictated by the network may serve to keep the vehicle on the road or to prevent it from colliding with nearby obstacles, depending on the type of sensor input and the driving situation that the network has been trained to handle.

FIGURE 12. ALVINN Architecture



ALVINN is trained by driving the Navlab over terrain of interest under control of a human driver. During recent months, ALVINN has resulted in some remarkable results, most notably high speed operation at velocities up to 62 miles per hour, and long distance robustness, demonstrated over continuous runs up to 22.4 miles in length.

6.2.5 Phase II and Demo B - Integrated Navigation, RSTA and Mission Planning

Phase II will culminate in Demo B, to be conducted in May 1993. Phase II will integrate basic map management functions including single vehicle mission planning, off-road semi-autonomous navigation, on-road semi-autonomous navigation, robust high-speed capability, landmark detection and identification, and rudimentary behaviors for following rules of the road.

1. The Surrogate Teleoperated Vehicle, built for Early User Test and Evaluation, will be fiber-optically linked to a single operator, with vision based tele-operation the principal mode of control. The Surrogate Semi-Autonomous Vehicle, built for Demo-II, will utilize the same chassis as the STV, but will replace the optional drivers compartment with a navigation sensor suite, and replace the fiber-optic payout system with computer rack carrying up to 60 VME Cards supporting on-board navigation, planning and control decisions.

6.2.6 Phase III and Demo C - Two Cooperating SSVs

Phase III will culminate in Demo C, to be conducted in October 1993. Phase II will integrate multiple vehicle control from a man-portable operator control unit, utilizing an updated map management and map building subsystem, which will rely on map information from overhead visual imagery, such as obtained from an unmanned aerial vehicle. Additionally, this phase will integrate the results of research in multiple agent interactions. Demo C will demonstrate two semi-autonomous vehicles executing coordinated navigation, including the use of the multiple vehicle operator control unit.

Demo C will also be utilized as an opportunity to evaluate the comparative merits of two emerging scalable parallel processors, the Intel/CMU iWarp, and the Hughes/UMass Image Understanding Architecture (IUA).

6.2.7 Phase IV and Demo II - Four Cooperating SSVs

Phase IV will culminate in OSD Joint Unmanned Ground Vehicle Demonstration II, to be conducted in October 1994. Phase IV will integrate a reconnaissance, surveillance and target acquisition subsystem, with technology derived from a synthesis of Image Understanding and Automatic Target Recognition efforts. This will result in a multiple vehicle mission subsystem with the capability of robustly navigating a team of four vehicles as a screening force. All SSVs will be equipped with the processor deemed most appropriate during Demo-C. The SSV team will be demonstrated in support of both offensive and defensive scenarios.

7.0 Conclusion

The Tactical Unmanned Ground Vehicle Program is a coordinated evaluation and development program with the objective of first fielding of an unmanned system by 1998. The TUGV program is being planned and managed with an awareness that it represents an initial step in the evolution and fielding of UGVs for combat applications and that its success or failure may have far-reaching consequences.

References

P. J. Besl, Range Image Sensors, Technical Report GMR-6090, General Motors Research laboratory, March 1988.

L. Davis, Stealth Terrain Navigation, these proceedings.

M.J. Hannah, A system for digital stereo image matching, *Photogrammetric Engineering and Remote Sensing*, 55(12):1765-1770, 1989.

M. Hebert, T. Kanade, and I. Kweon, 3-D vision techniques for autonomous vehicles, Technical report CMU-RI-TR-88-12, Carnegie Mellon University, August 1988.

K. Kluge and C.E.Thorpe, Explicit models for robot road following. In: C.E. Thorpe (ed) *Vision and navigation: The Carnegie Mellon Navlab*, Kluwer 1990.

L.H. Matthies, Stereo vision for planetary rovers: Stochastic modeling to near real-time implementation, Technical Report D-8131, Jet Propulsion Laboratory, Pasadena, CA, January 1991.

H.K. Nishihara, RTVS-3: Real-time binocular stereo and optical flow measurement system, System Description manuscript, Teleos, Palo Alto, CA, July 1990.

D.A. Pomerleau, J. Gowdy, and C.E. Thorpe, Combining Artificial Neural Networks and Symbolic Processing for Autonomous Robot Guidance, these proceedings.

D.A. Pomerleau, Neural Network Based Autonomous Navigation, In: *Vision and Navigation: The Carnegie Mellon NAVLAB*, Charles E. Thorpe (ed), Kluwer Academic 1990.

C. Weems, M. Herbordt, M. Scudder, J. Burrill, R. Lemer, and T. Williams, Status and Current Research in the Image Understanding Architecture Effort, these proceedings.

C. Weems, S. Levitan, A. Hanson, E. Riseman, J.G. Nash, and D. Shu, The Image Understanding Architecture, *International Journal of Computer Vision*, Vol. 2, No. 3, pp. 251-282, 1989.

RADIUS
The Government Viewpoint

Donald J. Gerson
Office of Research and Development

Abstract

The Office of Research and Development, with major involvement and support from the Defense Advanced Research Projects Agency (DARPA), has begun a highly applications-oriented project intended to provide Image Understanding (IU) technology in a fully and semi-automated support system of Human-Machine Interface (HMI) interactive tools to the Photo Interpreter (PI) and Imagery Analyst (IA). The central concept of RADIUS (Research and Development for Image Understanding Systems) is that of Model Supported Exploitation (MSE). Two- and/or three-dimensional site models are developed and/or maintained by analysts using imagery source data, imagery-derived information, and appropriate non-imagery sourced information (often called "collateral"). IU technology and necessary non-IU technology are used where feasible to integrate this base of information which can be accessed spatially via the now-developed site model and displayed or rendered in support of the IA during the imagery exploitation and reporting process. As new imagery is obtained, it may be registered to the site model, or through the site model to other images, to support the specific exploitation tasks and applications which will be developed. The current effort is the Concept Definition Phase. This phase will determine the viability of current technology to perform these tasks and will define future activities.

1 Introduction

An important goal within the Intelligence Community is to provide greater assistance to the IA in analysis of imagery, as well as in reporting (communicating) information which is derived during this process. The increasing availability and deployment of sensors capable of providing imagery source data in digital form, or the recent and expected near term technology advances, have created the environment and established the time to begin serious planning and research for capabilities, workstations, software, and tools that can better support the IA as he/she exploits multiple source image data.

The opportunities to assist in the derivation of information from softcopy imagery are boundless, and much has already been done. In RADIUS, we are attempting to move to a new plateau; one which will provide the IA a greater level of capability to manipulate both the imagery being analyzed and information already known about the site it represents. RADIUS will draw on previous efforts, and on many years of IU research, to produce this new capability.

The primary goal of RADIUS is to provide the IA a new and powerful approach with which to derive current information from imagery. This approach consists of modeling sites of interest and using these models in the imagery exploitation process. RADIUS is, therefore, focusing on model supported imagery exploitation.

2 Background

IAs use information from a variety of sources in exploiting imagery: information obtained from both imagery and non-imagery sources. This information often has positional or locational elements in common. In addition, many issues of importance to national security and defense policy can be related to a single site, a geographic area/region, or a collection of associated sites.

Informal site models are now employed in imagery analysis and presentation tasks in several forms and renderings: maps, annotated reference images, line drawings, physical small-scale models, and image perspective transformations. Models such as these, and far more formal models, can and will be used more widely in the imagery exploitation process, but only if the time and cost of building and maintaining them can be significantly reduced, and the quality of information maintained.

We believe that MSE, based on three dimensional models defined for sites of interest, has the potential for becoming a unifying concept for several elements of the imagery exploitation process. For example, MSE can support the development and use of reference aids, semantic structuring of databases, and development of imagery-derived products. Taken together, these elements represent a significant portion of the imagery exploitation process.

Most efforts to define and test concepts for integrating the tools used in softcopy imagery exploitation - textual and formatted databases and softcopy imagery exploita-

tion workstations – have recognized the need (though not explicitly) for a model-based approach. One project investigated ways of accessing "collateral" information via icons located on graphic representations of the images being exploited; and another developed a concept of tying relevant information, including exploitation requirements, to the image at hand. It also considered the possibility of expressing the information derived from the image as icons placed on the image and, if appropriate, generating a narrative report directly from this process. While these efforts demonstrated interactive methods for associating imagery with collateral information, approaches to directly assist the IA in exploiting current imagery are still needed.

3 Approach

RADIUS builds on these earlier efforts. It explicitly postulates site models as an essential element of any future softcopy exploitation operations concept. An MSE approach to softcopy exploitation has several benefits:

- It provides a common ground for communication between technologists interested in this aspect of IU and end users of the technology.
- It focuses these IU technology interests on an important national problem.
- It encourages system developers to see end-user data integration issues in terms recognizable by both the users and the technologists.

The approach is to take maximum advantage of the available imagery by improving analyst productivity, providing a more timely product, and potentially improving the quality of the product as well. The focus of the approach is to assist the IA in more effective use of imagery and other information that is available.

A significant IU research challenge is to find ways of using pre-knowledge about a site to help build a model of the site automatically. Such assistance might consist of existing informal models, such as maps or drawings, more formal representations of pre-knowledge such as structured databases, or knowledge in the analyst's head, which is used to guide the machine process. Use of this pre-knowledge may be essential in order to adapt existing IU techniques to the operational needs of imagery intelligence within the next decade.

The RADIUS challenge goes beyond that of an interesting IU problem. While IAs are by nature graphically oriented, and have always used informal models of the type noted above, we are just beginning to explore the concept of building, maintaining, and using more formal site models. The technology of site model building and an operations concept for the use of MSE are being developed concurrently. We have support from many organizations within the intelligence and imagery communities. Presently, the RADIUS contractor team and the U.S. Government have been working together in conducting surveys and experiments with IAs to assess their exploitation tasks, needs, and preferences. We will soon be investigating with analysts further definition of various concepts for HMI support to building/maintaining models and for rendering and displaying the results of the modeling process.

As part of this process, we will examine several specific applications of site models in the imagery exploitation process including:

- Registration, which in this context means the process of mathematical mapping of points on an image (from any type of sensor, possibly including ground photos) to points on a model. Currently, the registration process in softcopy requires selecting pairs of tie points (the same features on each image are identified by the user). This can be a time consuming task, especialy for certain sensors, and when integrating sensor data. Fully automated or highly automated registration has important assistance value to IAs in certain applications and for many exploitation tasks.
- Site Baselines, which are presently predominantly textual data, may have graphical components or depictions that are responsive to the exploitation requirements to which the IA is responding, and to the IA's understanding of these requirements and his/her individual needs, knowledge, and experience. Baselines narratively describe the general layout, certain objects and features, and, where appropriate, the table of organization and equipment (TO&E) found at this site. Baselines typically contain normalcy guidelines and overview sections where unusual activity or events that have been identified during the reporting period are noted and maintained. Subsequent remarks or comments added to it are responsive to the exploitation requirements describing information needs and will accrue over a period of time until updated or a new baseline is produced. This process provides one of several mechanisms to communicate historical and highly current information to the users of imagery-derived data, to support trend analysis, or to relate activity and change detected at this site which may be applicable elsewhere.
- Site Folders, which are reference aids to assist IAs in the exploitation of imagery of specific sites. These folders usually consist of imagery relating to a specific site. Maps, charts, drawings, messages, notes, and clippings from open literature are often included. Site folders assist IAs in performing analysis of changes and significant activity at these sites. These reference aids are particularly useful for IAs who are not very familiar with the site in question. This factor is particularly important when IAs must be able to exploit imagery from many sites quickly. Both Site Folders and Baselines exist today i hard copy. RADIUS site models will provide improved and integrated access to selected Site Baseline and Site Folder data which can be or will be maintained in various forms, predominantly digital, in the future MSE environment that is anticipated.
- Perspective, geometric modeling, and orientation are three closely related, overlapping concepts that have evolved from early experiments with proto-

types and other specialized equipment, and from operational experience with softcopy exploitation systems. This early experience shows promise in enhancing analysis techniques and in presenting and reporting analysis results. In brief, these concepts use imagery and models for the following capabilities. Perspective: "Show me what the site looks like from this new viewing position." Geometric modeling: "Show me what the site might look like under selected sensor acquisition parameters or environmental conditions." Orientation: "Help me compare this image of the site with another image (or model, or map) by matching positions on one with the other," (or simply, "put a north arrow on the image"). All three concepts share the need for processing multiple images and for occasionally producing 3D renderings of site models and/or registered images.

4 Programmatic Approach

RADIUS is intended to provide research results within five years that can be developed and deployed operationally within ten years. The current RADIUS project is Phase I of a potential three-phase effort. This first phase will last for two years, and will result in: 1) a decision regarding the level of maturity of IU for an operational application, and 2) a design for a demonstration/testbed system to be built during the second phase. Phase II will develop a system on which an IA can utilize MSE in a simulated operational context. During the later part of Phase II, detailed planning for a system design which would lead to a Phase III RADIUS-like MSE operational deployment will begin, so that funding decisions might be initiated for the development, transition, and integration of the RADIUS proven and cost-effective IU capabilities. If Phase II is successful, a decision will be made regarding a Phase III operational deployment. The Phase I/Phase II roadmap is depicted in Figure 1.

The Phase I RADIUS contract has two distinct components: the Concept Definition which will identify user needs and define the technical, performance and programmatic requirements for Phase II; and Technology Development, which will identify required technologies, determine their efficacy, determine required improvements, and either perform in-house research or define needed research to accomplish the improvements. There are also three concurrent efforts supporting RADIUS: 1) the RADIUS Common Development Environment (RCDE), 2) development of an experimental data set, and 3) three DARPA contracts for research supporting RADIUS technology development. The RCDE will provide a common environment for transfer and integration of all supporting technology during RADIUS Phase II. The RCDE contains the basic manual tools for building three dimensional models of terrain, buildings, and other man-made structures. It also will provide the necessary interfaces and "hooks" to add IU technology that more fully automates the human steps now necessary for building and maintaining site models, and for supporting selected IA exploitation tasks involving the application and use of such models. The RCDE is described in greater detail in a related paper in these proceedings [Mundy 92]. The experimental data set will be available for use by all participants in RADIUS-related efforts. This unclassified data set will consist of a substantial number of images, associated parameters, and supporting data which should usefully represent or characterize the imagery used and site complexity typical of many operational imagery exploitation environments.

A RADIUS Working Group has been formed to coordinate the RADIUS efforts discussed above. It meets bimonthly, and consists of key Government and contractor personnel participating in the RADIUS and RCDE projects. Researchers from the DARPA IU RADIUS-related studies will be added upon award of these contracts. The Working Group also provides an opportunity to report progress to those not directly participating in RADIUS, such as the DARPA IU Community, and representatives of potential Phase II contractors.

5 Conclusions

The Phase I RADIUS contract was awarded to Hughes Aircraft Company in July 1991. Subcontractors are BDM, Inc., Control Data Corporation, Hughes Research Laboratories, and the University of Southern California. The RCDE contract has been awarded to General Electric, with SRI International as a subcontractor. This team of seven contractors has been charged with accomplishing the RADIUS Phase I activities; their plans, approaches, and project status are being presented at this workshop or reported in other papers in these proceedings [Edwards 92].

References

J. Edwards, S. Gee, A Newman, R. Onishi, A. Parks, M. Sleeth, and F. Vilnrotter, "RADIUS: Research and Development for Image Understanding Systems – Phase I," *Proc. DARPA Image Understanding Workshop*, San Diego, CA, Jan. 1992, Morgan Kaufmann Publishers, San Mateo, CA 94403.

J.L.Mundy, R. Welty, L. Quam, T. Strat, W. Bremner, M. Horwedel, D. Hackett, and A. Hughes, "The RADIUS Common Development Environment," *Proc. DARPA Image Understanding Workshop*, San Diego, CA, Jan. 1992, Morgan Kaufmann Publishers, San Mateo, CA 94403.

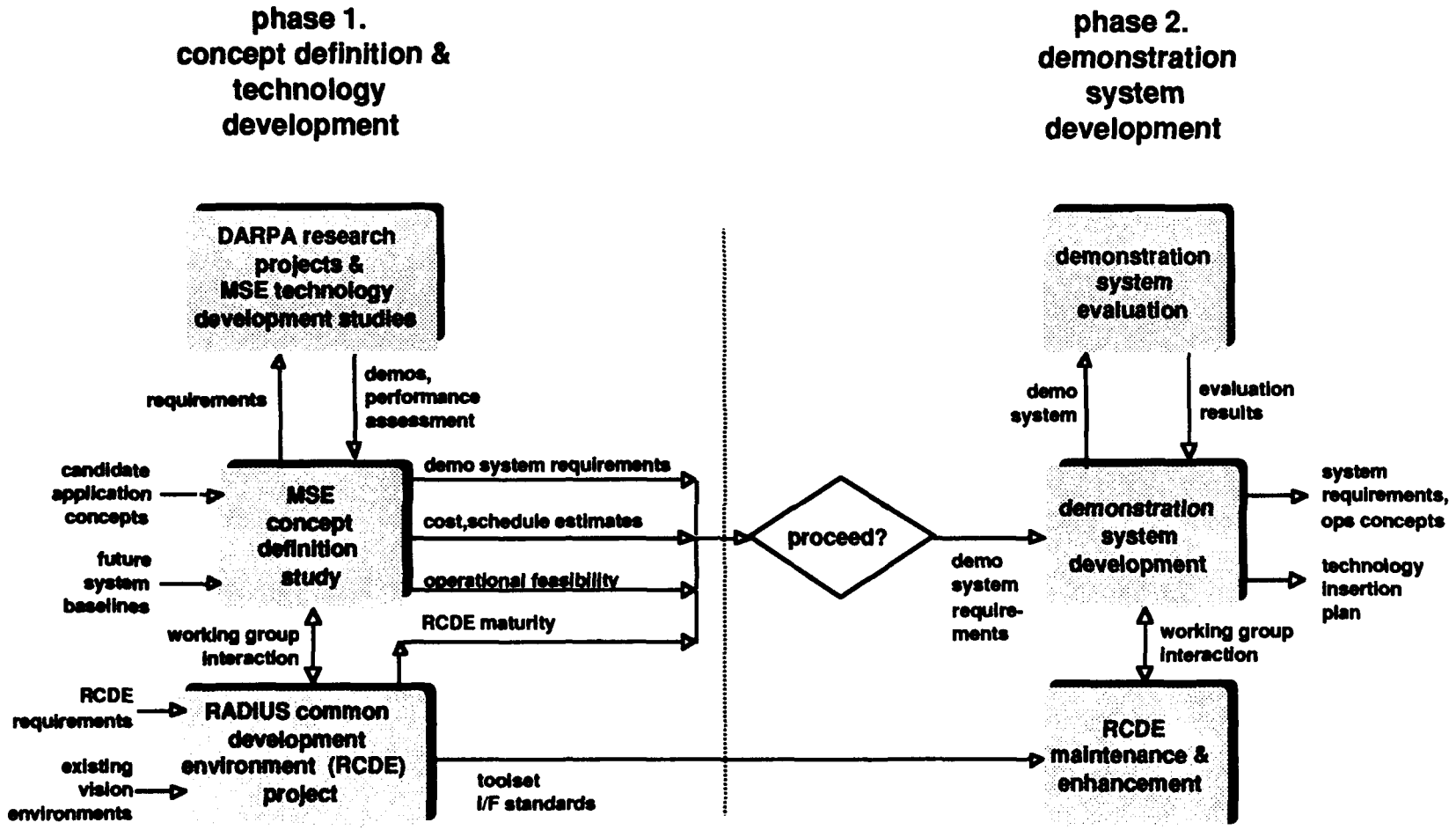


Figure 1. RADIUS Roadmap

RADIUS: Research and Development for Image Understanding Systems Phase 1

Jeffrey Edwards, Shirley Gee, Arthur Newman, Randall Onishi,
Alan Parks, Mark Sleeth and Felicia Vilnrotter
Information Systems Laboratory, Data Systems Division
Hughes Aircraft Company
El Segundo, CA 90245

Abstract

The Research and Development for Image Understanding Systems (RADIUS) project will focus IU research on the needs of operational imagery analysts (IAs). Model-supported exploitation (MSE) was selected by the government as the underlying concept to be validated, developed and demonstrated in RADIUS. IU-based tools for the creation and maintenance of site models are the core MSE technologies. Additional model-based image exploitation tools will also be examined under this contract.

Hughes Aircraft Company is teamed with BDM International, Inc. (BDM), McLean, Virginia; Control Data Corporation (CDC), Minneapolis, Minnesota; Hughes Research Labs (HRL), and the University of Southern California (USC). BDM is coordinating a series of experiments at the National Exploitation Laboratory (NEL); the other team members are involved in technology development.

1. Introduction

IAs are charged with providing accurate and timely information based on a detailed examination of image and collateral data. Collateral may take the form of reports derived from previous images, descriptions of the imaged site and normally observed activities, line drawings, maps and other textual or graphical materials. An accurate, geographically referenced 3D site model can serve to tie imagery and collateral together in a consistent, intuitive manner and provide uniform database access. In addition, site models are expected to support new analytic and presentation techniques, such as data fusion and animated reports. Site models will also support other IU-based exploitation tools that will act as intelligent assistants to the IA.

In selecting MSE as the focus for RADIUS, the government is acknowledging the maturity of IU and its potential to provide valuable assistance to an operational intelligence task. By the same token, the government anticipates that considerable application-oriented development will be required to integrate IU capabilities into an experimental testbed.

The goal of RADIUS is to demonstrate the viability of utilizing image understanding technology to meet the needs and desires of the imagery analyst community. RADIUS was conceived as a two-phased project to develop a demonstration system testbed that enables IAs to perform exploitation tasks using IU technology. After considerable analysis, the government selected MSE as the operational concept on which RADIUS would be based. The applications that will be included are Site Folders and Baselines; Registration; Perspective, Geometric Modeling & Orientation; and the capability to build and maintain site models to support these applications.

Phase 1 of RADIUS is devoted to the development of requirements and operations concepts for the testbed system and to characterizing, enhancing, and developing IU technology that is required to meet those requirements. Phase 2 of RADIUS will be devoted to developing the testbed system and continuing the research and development in IU technology required to meet the system requirements.

The following sections treat the various elements that must be considered in arriving at a testbed design; IA requirements, current IU capabilities, and the possibilities for technology development.

2. The IA's Perspective

RADIUS proposes the introduction of automated tools for creating, maintaining, and using site models as a means of increasing the productivity of the analyst community. Analysts must accommodate an evolving exploitation environment and mission, including increases in image volume, a rise in per-analyst workload, and new sensors providing unfamiliar data.

The introduction of MSE tools will require changes in the way analysts currently perform their duties. These changes are understandably points of concern for the individual analyst. While the promised benefits of MSE tools are large, the efforts required to initially implement and periodically maintain the associated site models can also appear large. It is important to emphasize that the analysts' greatest concern is to complete their assigned exploitation tasks in a timely and accurate fashion — a job which, from their point of view, has been achievable in the absence of site models and associated tools for years.

Accordingly, a primary goal of RADIUS must be to demonstrate to the analyst community that the increases in productivity resulting from site-model supported exploitation far outweigh the efforts invested to create and maintain the site models. Discussions and demonstrations must be framed within the context of a typical analyst's daily tasks and responsibilities. The following sections will summarize several overall exploitation functions currently performed by analysts, and will explore specific tasks targeted under RADIUS to provide automated support.

2.1. The IA's Tasks

While the analyst community is gradually moving toward softcopy imagery exploitation, the current analyst environment is still dominated by hardcopy imagery, text based reporting, and text based supporting information.

Within this environment, analysts conduct a broad variety of exploitation functions characterized by the amount of time available to produce an intelligence product. Time requirements, in turn, generally dictate the level of detail that the product can entail. *Phase 1* exploitation is most time critical; analysts typically review images of many different sites, therefore, they must rely on hardcopy maps and reports for rapid orientation to each site. The main Phase 1 task, *Indications & Warnings* (I&T), looks for evidence of activities that are of immediate concern for national security. *Phase 2* analysts have a more consistent set of sites for which they are responsible, have less severe time constraints, and examine imagery in greater detail. Their duties include *Search*, in which imagery covering large geographical area is examined for particular objects or activities, and *Surveillance*, in which a history of activity for a particular location is kept. *Phase 3* tasks focus on larger intelligence problems, such as trend analysis or the analysis of new equipment at a site (i.e., *Science & Technology* tasks). Textual readouts for examined imagery are stored in a shared database (NPIC Data System), while additional overall summaries and visual/graphical products are prepared and distributed off-line.

2.2. Model Supported Exploitation

The RADIUS RFP identified ten specific analyst activities amenable to automation. The list includes the Site Modeling task itself, the Site Model Update task, and eight exploitation tasks collectively identified as the MSE applications: Site Folders & Baselines; Registration; Perspective, Geometric Modeling & Orientation; Detection & Counting; Negation; Change Detection; Trend Analysis; and Recognition Guides. A subset of these applications was identified by the Hughes team as striking the best balance between applications offering the most payoff to the analyst community and project resources. The following four applications provide analysts a basic suite of tools for accessing static and previously extracted data, exploiting multiple images in a coordinated fashion, and visualizing sites with new degrees of freedom.

Site Modeling & Update — construction and maintenance of a 2D/3D site model from multiple images of a site.

Site Baselines & Folders — use of site model as a graphical front end to access databases of site information, collateral, history, functional descriptions; "best image" tutorials to acquaint analysts with an unfamiliar site.

Registration — matching corresponding parts between an image and a site model. Supports image-based access to databases; indirectly provides 3D image-to-image registration, tracking cursors, coordinated exploitation of multiple images.

Perspective, Geometric Modeling & Orientation — generation of synthetic imagery to depict a site from arbitrary view points, provide fly-throughs and calculate time-lapse sequences.

These four applications provides a significant package of capabilities for model-supported image exploitation. The other MSE applications are considered to be directly dependent upon the capabilities provided by this set. Additional research or focused development can build upon this foundation of generic capabilities to provide more highly automated tools addressing the other applications.

2.3. The Successful Introduction of MSE

The question of whether or not the MSE Concept is useful and desirable is directly related to the IA's assessment of its payoff relative to the amount of effort required to initiate, maintain, and use the provided capabilities. The exploitation community

believes that analysts are hired primarily for exploiting imagery, not building and updating site models. Analysts may be motivated to perform these latter functions as well, if the resulting benefits permit them to perform their primary duties better and/or more quickly. A successful RADIUS Phase 1, leading to a Phase 2 testbed system, is dependent upon the developers identifying and validating technologies that can demonstrably show a path toward a rational and consistent level of interaction between analysts and automated tools.

3. The DARPA IU Research Community Role

DARPA's role in the RADIUS project is important both from the standpoint of DARPA's role as co-sponsor of the program and as the sponsor of IU projects which have pushed the envelope of IU technology to its current level of maturity. Much of the expertise for advancing the areas of IU technology needed by RADIUS resides in the DARPA research community.

IU research spans the spectrum from being highly theoretical, where results are analytical in nature, to being application specific, where a well-defined capability is shown to exist over a well-specified range of domain imagery. Developed technologies may be tested on anywhere from a few available images to a fairly complete set of images showing a full range of image acquisition and domain conditions. These technologies range from the algorithm to the system level, and support semi- to fully- automated functional capabilities.

Most IU research was performed independent of knowledge of RADIUS needs, and without exposure to, or use of, RADIUS application imagery or its unclassified equivalent. Therefore, one of the first tasks to perform during the initial phase of RADIUS is the identification of those technologies which can support RADIUS capability needs and the characterization of the range of conditions under which these technologies function. As a result of this process, potential gaps and/or shortfalls in IU technology capabilities, *in terms of RADIUS technology needs*, may be uncovered.

With initiatives such as the DARPA Image Understanding Environment (IUE) project, work developed at different institutions will be able to be exercised and combined relatively easily. (See the paper by Mundy, et al, "The DARPA Image Understanding Environment Project", in this proceedings.) Since no such standard environment or methodology currently exists, different systems will need to be exercised on different platforms, and data formats may need to be changed when coupling of algorithms is required.

DARPA continues to sponsor IU technology projects, including RADIUS related IU research. As gaps and shortfalls are recognized, information will be fed to DARPA. IU research efforts may be used to help fill these technology gaps, overcome technology shortfalls, or support development of application concepts not being addressed under the current RADIUS contract. Hughes will work with research institutions during RADIUS Phase 1 to ensure a smooth transfer of technology.

4. The Contractor's Perspective

The goal of RADIUS is to integrate the needs and desires of the image analyst community with the research performed by the IU community. Coordination of these non-aligned interests is considered a keystone to project success. From a system engineering standpoint, intimate knowledge of analyst activities and priorities is vital to designing a useful testbed system. From a technological perspective, access to, and incorporation of leading-edge developments is critical to designing a powerful and robust system.

4.1. Outcome of RADIUS Phase 1

The major goal of RADIUS Phase 1 is to develop requirements for the testbed system that will be developed during the second phase of the project. A significant number of the requirements must use IU technology as a solution.

With limited time and budget, resources must be applied to proper areas of research to ensure success. Therefore, during Phase 1 the project will first look at existing technologies that address the RADIUS problems.

4.2. Methodology

The large-scale evaluation and gradual optimization (LEGO) methodology being used on RADIUS was designed to accommodate the time and level-of-effort constraints inherent to research projects. It assumes that maximal return on investment is realized when work is focused on the enhancement of applicable pre-existing technologies and the development of new capabilities where required.

The first phase of LEGO requires a large-scale evaluation of both system needs and the current state of research technology. The MSE tasks selected for RADIUS Phase 1 will be decomposed to their functional components, making a comparison with existing IU technology possible. IU systems will be mapped onto the functional components to determine potential areas for MSE automation, as well as those areas for which no IU technology exists. In addition,

the IU systems will be characterized to identify their operational range of conditions.

RADIUS functional requirements will be identified through a series of experiments being conducted with image analysts at the NEL. The extent to which these requirements are not met by the characterized systems will motivate technology enhancement and development. Technology enhancement and development proposals will be prioritized based on their importance to IAs, as determined by the concept validation experiments, and their potential for realization within the RADIUS time frame.

4.3. RADIUS Related IU Research Community Work

In this section we include the preliminary list of technologies we believe can support needed RADIUS capabilities, and how they map into the structure referred to in the previous section. This is by no means a final or complete list and we encourage the community to help us to identify other work which provides needed capabilities. Likewise, we would appreciate as much feedback as possible with respect to the technologies we included in order to help us to identify how best they fit into the RADIUS puzzle.

4.3.1. Registration Application

Registration Area Reduction:

HAC	SCORPIUS	Cloud Detection

Model-to-Image Registration:

GE	Thompson & Mundy	Extended Vertex Pair Matching
GE	Nguyen, Mundy & Kapur	Constraint System
SRI	Barrow, Tenenbaum, Bolles and Wolf	Chamfer Matching
HAC	SCORPIUS	Image to 2D Model
USC	Medioni & Nevatia	Image to Map - linear features
HRL	Silberberg, Harwood & Davis	Image to 3D Model

4.3.2. Site Modeling Application:

Image-to-Image Registration:

CMU	Perlant & McKeown	Coarse Registration via spatial database, Fine Registration using automatic selection of control points (e.g., Shadow Corners)
USC	Medioni & Nevatia	Registration using linear features
HRL	Zikan & Silberberg	Local feature based matching

Object Modeling:

Broad Area Delineation:

CMU	McKeown & McDermott	Rule-based segmentation and interpretation system for Airports
Region-Based Segmentation		Methodologies based upon texture, intensity, etc...

Building Detection and Delineation:

CMU	Irvin & McKeown	BABE, Shade, Grouper, Shave,...
USC	Huertas & Nevatia	Building Detection using Shadows
USC	Mohan & Nevatia	Rectangular roof detection and stereo matching
CMU	Herman & Kanade	3D Mosaic Scene Understanding System
U of M	Venkateswar & Chellappa	Building Detection using Shadows
SUNY	Liow & Pavlidis	Building Extraction using Shadows
U of M	Hwang, Davis & Matsuyama	2D detection of houses and roads

Roads and RRs:

CMU	McKeown	Road Network Finder
HAC	Gee	Corridor Finder
SRI	Fischler, Tenenbaum, & Wolf	Roads and RRs
HRL	Silberberg, Kim & Olin	Road Detection
U of M	Canning, Kim, Netanyahu & Rosenfeld	Road Detection
U of M	Hwang, Davis & Matsuyama	Detection of Roads and Houses

Runways and Taxiways:

CMU	McKeown & McDermott	SPAM
USC	Huertas, Cole & Nevatia	Runway detection

Ports:

USC	Huertas	Piers

Terrain Elevations:

CMU	Hsieh, Perlant & McKeown	S1 and S2 Stereo Systems
USC	Cochran & Medioni	Area/Feature Based Stereo System
SRI	Hannah	Stereosys
SRI	Barnard	Cyclops Stereo System
GD	Stereo System	
MIT	Grimson	Feature Based Stereo

4.4. Characterization of Existing Technology

The major objective of the system characterization task is to provide the RADIUS community with performance measures of various image understanding systems. These characterizations will be used as input to determine technology shortfalls with respect to RADIUS system requirements. A set of metrics will be determined for each class of system whose performance is to be characterized. For example, road finders would form one class of systems, with percent of roads found forming one of the performance metrics.

To support testing of IU systems in unclassified environments, a set of model board and aerial images will be provided by the government. The intent of the test set is to provide numerous examples covering a variety of site types, plus a range of natural conditions such as clouds, haze, textured areas and water.

In most cases, the developer of the program will perform the first set of performance tests. In cooperation with the government, a mechanism will be established to distribute the unclassified imagery set to interested IU developers. Based on the level of performance achieved on the unclassified test set, the system will then be ported to a classified platform for further tests against classified imagery. A final report of the system's performance characterization will be made for each system tested. The system's developers will be kept involved in the entire testing process. In selected cases, RADIUS personnel may visit the developer for orientation and training on their system. As testing progresses, the developer will be consulted for help with parameter setting, problems, and advice as to what is required to optimize system performance.

4.5. Development of Requirements — IA Experiments

In offering automated tools for image exploitation, it is necessary that the customer and developer specify the capabilities such tools will provide and the requirements they will meet. While these requirements define the scope of the work to be done, perhaps more importantly, the development of such requirements also offers an opportunity to validate that the proposed tools will indeed provide capabilities that are useful and amenable to the end user. As one of its main goals, RADIUS will consult the analyst community as a sounding board to validate proposed areas of technology development.

A formal series of Imagery Analyst experiments will be coordinated by BDM to be performed at the National Exploitation Laboratory (NEL). The experiments will seek to identify and characterize key IA issues and concerns regarding the use of proposed MSE tools. Results from these experiments will be incorporated into the Phase 2 system requirements document and will help to determine which IU technologies will be developed.

4.5.1. Sites and Site Objects

MSE tools developed for imagery analysts must be designed in light of the types of objects that IAs will require in site models. The required content for a site model will vary; it is clearly influence by the geographical location of a particular site, the activities and objects typical at that site, the intelligence significance of that site, and the nature of the exploitation efforts that will be directed against the site.

Characterizing site model content has important ramifications for the technologies that are being pursued under RADIUS. Site modeling tools should be capable of operating on as many types of objects as possible to reduce the amount of manual modeling and correction. A preliminary survey of the 6 RADIUS sites identified a large number of terrain types, object classes, object densities, weather conditions and viewing geometries. While only some objects at a site will be modeled, all objects visible at the site contribute to the overall complexity of the image being exploited; all of the objects produce signatures and features that automated IU algorithms must sift through in identifying the significant objects. Note that most imagery used by IU developers does not reflect the complexities evident in the customer's typical operational imagery. It is important then, for developers of RADIUS related IU to consider the effect of differences between this operational imagery and any test imagery they are using to validate their work.

The required content of a site model is likely to be augmented to include objects other than those required by analysts. Such objects will include, for example, those required by the IU algorithms making up the automated exploitation tools. The Registration application is a good example of this; algorithms may require that various insignificant, large scale objects be added to the model to expedite

an initial coarse registration. Even when refining the registration with smaller objects, the algorithms may require that specific objects with strong signatures be included in the model despite their unimportance.

A primary goal of the first NEL analyst experiments will be to assess the specific types of object classes that analysts want to see in site models. The results will aid in prioritizing the types of IU systems that will be pursued for validation under Phase 1.

4.5.2. Accuracy and Level of Detail

Related to the question of what objects should be in a site model is the question of how accurately they must be modeled and to what level of detail. As before, the tradeoff here is between the quality of an IU product (e.g., a site model or a registration result) and the resources required to deliver that quality of product. An analysis of the latter must assess the likelihood of successfully developing the required IU technology, as well as the cost and time required to do so. Other related factors, considered less significant by the customer at this early stage of the project, are the predicted run-times of the automated tools and the hardware/software environment required to support them.

Another facet of the first NEL experiment will seek to investigate these issues. Imagery analysts will be formally consulted to assess the level of accuracy and level of detail that they need when constructing site models and using them with associated automated capabilities. Note, however, that the final requirements for accuracy and detail are likely to continue to evolve as the program progresses. It is anticipated, for example, that integration and testing of Phase 2 testbed components will bring out and motivate additional issues that will refine requirements.

4.5.3. Analyst Interface

As previously discussed, analysts' acceptance of Site Modeling and the MSE Applications will, to a large extent, be determined by their perception of whether or not there is substantial benefit in using and maintaining these capabilities. An important goal of RADIUS will be to assess what role an analyst is willing to assume in constructing, maintaining, and using site models.

Some issues coming out of this assessment are likely to be motivated by human psychology; some may appear irrational or contradictory from the view of technology developers. For example, previous analyst interface research has revealed a dislike for the manual selection of tie-points for the purpose of image-to-image registration. Based upon this result, it is likely that this process will have to be totally automated in order to be accepted. If, on the other hand, an automated registration tool can initially align a site model to an image with a relatively "good" accuracy, analysts may be willing to make minor translational corrections to fine tune it, thus relieving the automation requirements of this task.

The second NEL experiment will investigate analysts' preferences in interfacing to automated exploitation tools. A goal will be to assess their willingness to assist automated tools in the task of site modeling; the types and amounts of information an analyst is willing to supply will be explored.

4.6. Strawman System Requirements and Capabilities

Requirements for the RADIUS Phase 2 testbed will evolve out of a compromise between IU capabilities and IA requirements. While the IAs may desire a completely automatic system to generate site models, they would probably not be willing to accept the level of performance for such a system, given the current state of technology. Assuming a semi-automated system, a balance must be struck between the level of performance IU can provide and the level of interaction required of the IA.

4.6.1. Quality of IA Interaction

One ingredient of this balance involves the perceived role of the IA in the site modeling process. On the one hand, the IA can have extensive interaction with an IU system during the modeling process, providing it with sufficient information so that its results require minimal correction or editing. On the other hand, highly automated systems may keep IA interaction to a minimum, leaving the IA to monitor the final output and correct it as needed. We anticipate that an IA's preference, as recorded during the IA experiments, will depend on the error rate of the system; the IA would prefer a more interactive approach rather than spend much time correcting errors.

4.6.2. Targeted Level of Automation

With these issues in mind, the strawman level of automation draws from both styles of interaction. The strawman defines a pattern of interaction for the IA that is consistent across the various site modeling tasks. It also takes a conservative view of the IU capabilities available to solve the range of site modeling tasks. This estimate is not intended to place an upper limit on the level of automation made available to a site model builder. It is intended to focus attention on the issue of consistency of user interaction, an important factor in IA acceptance.

The strawman level of automation requires the user to specify some attributes of the object to be modeled, such as its shape and where that object is in the image. For each site model object, the IA selects a geometric primitive. Simple, frequently encountered shapes will be represented within the site modeling tool. For complex shapes, a specification tool to compose custom primitives will be made available to the IA. Upon selecting the shape, the IA delineates the image area within which the site object exists. Once the geometric primitive and image area are specified, the system automatically determines the orientation and scale of the site object, and returns the parameters that reflect the optimal fit between site object and geometric primitive. This result is graphically displayed to the IA, who may then correct the fit, embellish it with additional detail, or attach semantic information.

We expect that this strawman would apply to a large subset of the complete range of image conditions and site object types. Optimal image conditions and site objects composed of simple geometric shapes are probably amenable to a more completely automated approach, while maintaining the required level of performance. More complex conditions would, conversely, require more IA interaction. Therefore, we may expect several approaches to modeling the same site objects to be included within any deployed system.

4.7. Technology Development

The identification of technology areas to enhance or develop follows from the decomposition of MSE tasks, the characterization of existing technology and the determination of IA requirements. Whether or not any of the characterized systems meet each of the application requirements will be determined. If any requirement is inadequately met there is a shortfall. When it is not met at all there is a gap.

RADIUS must next make recommendations concerning the course of action to be followed with respect to each shortfall and gap. A cost / benefit analysis will be performed based on criteria such as importance to IAs, frequency of occurrence during MSE tasks, implications to other MSE applications, availability of work arounds and cost. Determining the answers to some of these questions will prove to be difficult, particularly cost and risk. For this reason, the DARPA researchers, especially those whose work is applicable to the MSE tasks, will be enlisted to help answer these questions on every occasion possible. Upon sponsor approval of the selected technology development areas, Hughes will then make recommendations as to who should pursue each enhancement. This will be based on familiarity with the technology, availability of people and equipment, cost effectiveness, and whether they are currently under contract.

5. Summary

Our approach to reaching Phase 1 goals includes concept validation experiments to determine critical IA requirements, IU system characterizations to assess the maturity of key technologies, and the identification of critical areas of IU research to be pursued. We look forward to working cooperatively with the IU community to accomplish the RADIUS Phase 1 goals.

Acknowledgements

The authors would like to thank Teresa Silberberg (HRL), Ram Nevatia (USC), Mike Kelly (BDM) and Mike Murphy (CDC) for their contributions to this paper.

RADIUS research is being supported by the Defense Advanced Research Projects Agency (DARPA), and the Office of Research and Development (ORD). Other organizations, such as the National Exploitation Laboratories (NEL), have expressed their interest in and support of the project.

References

Barnard, Stephen T. 1990. "Recent Progress in CYCLOPS: A System for Stereo Cartography." Proceedings of the DARPA Image Understanding Workshop, Pittsburgh, Pennsylvania, (September): 449-455.

Barrow, H.G., Tenenbaum, J.M., Bolles, R.C., and H.C. Wolf. 1977. "Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching." Proceedings of the ARPA Image Understanding Workshop, (April).

Bogdanowicz, Julius. F., and Newman, Arthur M. 1989. "Overview of the SCORPIUS Program." Proceedings of the DARPA Image Understanding Workshop (May): 298-308.

Canning, J., Kim, J.J., Netanyahu, N., and A. Rosenfeld. 1988. "Symbolic Pixel Labeling for Curvilinear Feature Detection." Proceedings of the DARPA Image Understanding Workshop, Cambridge, Massachusetts, (April): 1080-1090.

Cochran, S.D., and G. Medioni. 1989. "Accurate Surface Description From Binocular Stereo." Proceedings of the DARPA Image Understanding Workshop, Palo Alto, California, (May): 857-869.

Fischler, M., Tenenbaum, J., and H. Wolf. 1981. "Detection of Roads and Linear Structures in Low-Resolution Aerial Imagery Using a Multi-Source Knowledge Integrating Technique." Computer Graphics and Image Processing, 15, 201-223.

Grimson, W.E.L. 1985. "Computational Experiments with a Feature Based Stereo Algorithm." IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-7(1), (January): 17-34.

Hannah, M.J. 1988. "Test Results from SRI's Stereo System." Proceedings of the DARPA Image Understanding Workshop, Cambridge, Massachusetts, (April): 740-744.

Hannah, M.J. 1985. "SRI's Baseline Stereo System." Proceedings of the DARPA Image understanding Workshop, Miami Beach, (December): 149-155.

Herman, M., and T. Kanade. 1986. "Incremental Reconstruction of 3D Scenes From Multiple Complex Images." Artificial Intelligence 30, 289-341.

Herman, M., and T.Kanade. 1984. "The 3-D MOSAIC Scene Understanding System: Incremental Reconstruction of 3-D Scenes From Complex Images." Proceedings of the DARPA Image Understanding Workshop, 137-148.

Hsieh, Y.C., F.Perlant, and D.M. McKeown. 1990. "Recovering 3D Information From Complex Aerial Images." Proceedings of the DARPA Image Understanding Workshop, Pittsburgh, Pennsylvania, (September): 670-691.

Huertas, A., Cole, W., and R. Nevatia. 1990. "Detecting Runways in Complex Airport Scenes." Computer Vision, Graphics, and Image Processing 51, 107-145.

Huertas, A., and R. Nevatia. 1988. "Detecting Buildings in Aerial Images." Computer Vision, Graphics, and Image Processing 41(2), (February): 131-152.

Hwang, V.S.S, Davis, L.S., and T. Matsuyama. 1986. "Hypothesis Integration in Image Understanding Systems." Computer Vision, Graphics, and Image Processing 36, 321-371.

Irvin, R.B., and D.M. McKeown. 1989. "Methods for Exploiting the Relationship Between Buildings and Their Shadows in Aerial Imagery." IEEE Transactions on Systems, Man, and Cybernetics, SMC-19(6), (November/December): 1564-1575.

Liow, Y.T., and T. Pavlidis. 1990. "Use of Shadows for Extracting Buildings in Aerial Images." Computer Vision, Graphics, and Image Processing 49, 242-277.

D.M. McKeown. 1990. "Toward Automatic Cartographic Feature Extraction." NATO ASI Series, Vol. F65, Mapping and Spatial Modelling for Navigation, edited by L.F. Pau, Berlin: Springer-Verlag, 149-180.

McKeown, D.M. and J. McDermott. 1983. "Toward Expert Systems for Photo Interpretation." IEEE 1983 Proceeding of Trends and Applications, 33-39.

Medioni, G., and R. Nevatia. 1984. "Matching Images Using Linear Features." IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6(6), (November): 675-685.

Mohan, R., and R. Nevatia. 1989. "Using Perceptual Organization to Extract 3-D Structures." IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-11(11), (November): 1121-1139.

Mundy, J.L., et al. 1992. "The DARPA Image Understanding Environment Project." Proceedings of the DARPA Image Understanding Workshop, San Diego, California, (January).

Nevatia, R., Price, K. and Medioni, G. 1990. "USC Image Understanding Research: 1989-1990." Proceedings of the DARPA Image Understanding Workshop, Pittsburgh, Pennsylvania, (September): 43-66.

Nguyen, V., Mundy, J.L., and D. Kapur. 1990. "Modeling Polyhedra by Constraints." Proceedings of the DARPA Image Understanding Workshop, Pittsburgh, Pennsylvania, (September): 515-529.

Perlant, F.P. and D.M. McKeown. 1990. "Scene Registration in Aerial Image Analysis." Photogrammetric Engineering and Remote Sensing, 56(4), (April): 481-493.

SCORPIUS Program Summary for 1989.

SCORPIUS Final Report (1990).

Silberberg, T.M., Harwood, D.A., and L.S. Davis. 1986. "Object Recognition Using Oriented Model Points." Computer Vision, Graphics, and Image Processing 35, 47-71.

Silberberg, T.M., Kim, J.H., and K.E. Olin. 1987. "Investigation in Context Cueing: Phase II" Final Report DAAK21-84-C-0131.

Thompson, D.W. and J.L. Mundy. 1987. "Three-Dimensional Model Matching From an Unconstrained Viewpoint." IEEE Int. Conf. on Robotics and Automation, 208-220.

Venkateswar, V. and R. Chellappa. 1990. "A Framework for Interpretation of Aerial Images." ICPR-A , 204-206.

Zikan, K., and T.M. Silberberg. 1988. "The Frobenius Metric in Image Registration." Technical Report SOL 88-13, Department of Operations Research, Stanford University, (July).

The Image Understanding Environments Program

J. Mundy*
GE Corporate Research and Development
Schenectady NY 94403

T. Binford
Stanford University
Stanford CA 94305

T. Boult
Columbia Univ.
NYC NY 10027

A. Hanson and R. Beveridge
Univ. of Mass.
Amherst MA 01003

R. Haralick and V. Ramesh
University of Washington
Seattle WA 98196

C. Kohl
Amerinex
Artificial Intelligence, Inc.
Amherst MA 01002

D. Lawton
Georgia Institute of Technology
Atlanta GA 30332

D. Morgan
Advanced Decision Systems
Mountain View CA 94043

K. Price
University of Southern California
Los Angeles CA 90089

T. Strat
SRI International
Menlo Park CA 94025

Abstract

The Image Understanding Environment(IUE) project is a five year program, sponsored by DARPA, to develop a common software environment for the development of algorithms and application systems. This paper reviews the history of the project and provides an overview of some the data structures which are currently evolving as a specification for the IUE. The ultimate goal of the project is to provide the basic data structures and algorithms which are required to carry out state of the art research in image understanding.

1 Introduction

1.1 The Nature of IU Research Software

Image understanding research is typically carried out by individual contributors who develop a specialized software environment for implementing and evaluating new approaches and concepts. In most cases, the software environment is thrown away as new approaches are considered or the researcher moves to a new research problem or new hardware platforms are introduced. A clear example of the latter case is the decline of the Symbolics Lisp Machine as a platform for rapid prototyping of IU algorithms.

This volatility of research software is not particular to image understanding research, but is perhaps characteristic of software systems in general. These systems evolve rapidly with time as new languages, programming techniques and hardware platforms emerge. However, this volatility and diversity places a heavy toll on the efficiency of the image understanding research community. It is difficult to share and evaluate new research ideas because they can be demonstrated only in the specialized environments generated by the originator of the idea. A steep learning curve is encountered in any attempt to acquire and operate these specialized and often fragile environments.

Another major source of inefficiency is that most IU experiments require an extensive set of rather standard algorithms and data structures to reach the level of processing and feature extraction upon which the new ideas can be tried. Typical examples are image smoothing, edge detection, curve fitting, feature grouping and camera calibration. Since each environment is somewhat different in its design of the basic data structures, it is necessary for the algorithm developer to reimplement the basic IU infrastucture in order to evaluate a complex algorithm. This process is repeated over and over at dozens of IU research laboratories each year as new re-

*The committee effort has been funded by numerous DARPA grants and associated funding partners under contract to each institution.

search projects start.

Finally, it is now widely recognized that significant applications of IU research can only be realized in the context of large systems. Examples of such applications are the UGV(Unmaned Guided Vehicle) project which will exploit image understanding for navigation and surveillance tasks, and the RADIUS(Research and Development for Image Understanding) project which is focused on the application of Image Understanding to photointerpretation. These extensive application projects cannot be realized without a common software environment to provide the integration of diverse system components, developed at different research institutions.

1.2 Initiation of the IUE

In late 1989, Rand Waltzman of DARPA, then manager for Image Understanding programs conceived and developed a new program called I4US. The decoding of this acronym is Intelligent Integrated Interactive Image Understanding. The name has since been shortened to IUE, for Image Understanding Environment.

The IUE program was announced at a meeting for DARPA Principal Investigators in Scottsdale, Arizona at the end of February, 1990. The project goal, as announced by Rand, was a five year program to design and implement a common software environment for the development and demonstration of image understanding algorithms and techniques.

At the time of the meeting, there were three environments which had reached a reasonable maturity and had attracted a sufficient number of users to demonstrate that the idea of an IUE was feasible. These three systems and general characteristics are summarized below.

- The Cartographic Modeling Environment(CME) has been developed by Lynn Quam of SRI International over the past decade [1]. The focus of CME is the efficient handing of large images and the representation of configurations of 3D object models on the earth's surface under perspective viewing. in support of site modeling for cartography and photoreconnaissance. CME is implemented in Lisp on the Symbolics Lisp Machine. CME is currently playing a central role in the RADIUS program as a proposed development environment for RADIUS experiments. This development environment is called RCDE, or RADIUS Common Development Environment.

  CME is currently being ported to the SUN-UNIX platform under X-Windows and Common Lisp.

- KB Vision (Knowledge-Based Vision) is a product of Amerinex Artificial Intelligence Inc., and is losely based on the VISIONS system developed at the University of Massachusetts at Amherst over the last fifteen years [3]. KB Vision combines Lisp and C components. C is used primarily for numerical and image feature processing tasks while Lisp is for high level reasoning about image content. KB Vision is widely distributed among users of image understanding technology and provides an effective interface for developing image feature segmentation and feature grouping algorithms. KB vision is currently being extended to provide the programming environment for the Image Understanding Architecture or IUA, a highly parallel, multi-granularity design.

- Power Vision was developed by Advanced Decision Systems(ADS), now a division of Booz, Allen and Hamilton, starting about 1986 [2]. Power Vision provides an object-oriented programming environment in Symbolics Flavors for a wide range of Image Understanding data structures. An effective user interface has been developed for displaying graphical interpretations of relationships between image features. Power Vision was used by ADS in many of their application studies, including their work on the Autonomous Land Vehicle Project(ALV).

These systems provided the conceptual basis for the IUE, but the scope of the IUE project is much broader than any of these existing systems. The goal is to provide an environment which can cover the full spectrum of IU research and support both C and Lisp application development.

To initiate the project, Rand Waltzman convened and chaired three meetings during the 1990-1991 period to develop an consensus in the IU community about the requirements for the IUE [1]. A number of teams were established to suggest specific application scenarios and propose skeleton architectures for the IUE. In April 1991, these team reports were reviewed, and the IUE committee was formed from representatives of each team. Since April, the IUE committee has met five times and has produced a requirements and design specification for the IUE. The design consists of over a hundred classes at the time of this writing and over 400 pages are required to document the classes! The extensive nature of the design illustrates one facet of the complex nature of the Image Understanding problem.

2 The IUE Program Schedule

The current IUE schedule is summarized in figure 1. The draft IUE specification was distributed for review in December 1991 to a review board consisting of experienced IU system developers and researchers for comment. The responses will be incorporated in the a revised document This final version will be broadly distributed in February 1992. At the same time, an RFP will be issued to select an integrating contractor for the IUE. The actual software development is to be carried out at a number of IU research laboratories under the coordination of the integrating contractor who will supervise the coding standards and provide the configuration control and documentation of the system. It is hoped that many university course activities will be initiated to evaluate the IUE design and provide code for the system[2].

It is expected that the IUE will be developed over a two year time period and the initial release will be

[1] The IUE project is currently monitored by Oscar Firschein who replaced Rand Waltzman as the IU program manager at DARPA in June 1991.

[2] Keith Price of USC is already experimenting with the draft specification as a means for supervising algorithm development by new graduate students in IU.

available at the end of 1994. It is planned to broadly distribute the IUE to IU researchers throughout the US at that time. The system will continue to be maintained by DARPA until the end of 1996. At that point, it is assumed that the IUE will be fully supported by the user community through maintenance charges and technical support fees. The IUE Committee will stay in existence during the same time frame to provide design guidance and to develop an IUE standards process.

The following sections provide an overview of the scope of the IUE and a summary of the draft object hierarchies which will guide the development of the system.

3 Scope of the IUE

The primary purpose of the Image Understanding Environment(IUE) is to facilitate exchange of research results within the IU community. The IUE will provide a platform for various demonstrations and tools for DARPA applications. These demonstrations and tools will become a primary channel for IU technology transfer. The IUE will also serve as a conceptual standard for IU data models and algorithms. The availability of standard implementations for basic IU algorithms will facilitate performance evaluation of new techniques and to track progress in algorithm improvements. The IUE is designed to support significant evolution of IU approaches and an effective programming environment for rapid prototyping.

The IUE is not intended to be a real time system although tools will be provided for the simulation of real time applications such as navigation. The IUE will not support special hardware accelerators but a standard image processing interface will be provided. There is no intention to generate a design suitable for embedding in larger systems, although object class components can certainly be used in the construction of new systems.

An important aspect of the development of the IUE is the support of various application scenarios. The following application areas have been selected to guide the design of the IUE environment.

- **Photo-interpretation** - Support the analysis of large aerial images with a variety of sensor modes including Electro-Optical(EO), Synthetic Aperture Radar(SAR) and Multi-Spectral(IR) images. The main tasks include image registration and fusion, object recognition, change detection and mensuration.
- **Smart Weapons** - Provide target identification and tracking algorithms to demonstrate the application of IU techniques in missile guidance. Application tasks include image sequences, active vision and object recognition.
- **Cartography** - Provide tools for the construction of cartographic databases from images. Application tasks include stereo, camera modeling and automated image feature extraction.
- **Visual Navigation** - Support ongoing development in land reconnaissance vehicles. Application tasks include, road tracking, terrain analysis, obstacle avoidance, and object recognition.
- **Industrial Vision** - It is anticipated that DARPA will form a new thrust in design and manufacturing technology within the the time frame of the IUE development. Application tasks in this area include, range image sensor development and range data analysis, image segmentation, automated model learning and visual feedback.

4 Design Principles

Object-Oriented

The central approach to the design of the IUE is the use of object-oriented design principles. Briefly, an object is a data structure with associated operations, or methods, which are naturally defined for the particular structure. The design is specified in terms of an object class hierarchy which represents abstraction relations between classes. For example a *"T" junction* is a special type of *junction*. The internal details of an object are hidden as much as possible so that new implementations can be "plugged in" without significantly affecting the rest of the system.

Standard Workstation Environment

The IUE will be built using off-the-shelf workstation components as much as possible to reduce development cost. The system will use the UNIX operating system and the user interface will be built on X-Windows. Standard widgit toolkits, programming environments and debugging tools will be used in the system even if license fees are involved. In addition, source code may not be available for all of these support components.

Languages

The system will support LISP and C by providing parallel object class hierarchies and a mechanism for communicating between the two language environments. Common Lisp Object System (CLOS) and C++ are the object-oriented language standards used in the IUE.

Interactivity

The IUE will make extensive use of graphical interaction to support the examination of features and to provide convenient tools for model construction, recognition and etc. These tools will be constructed within a uniform user interface methodology and will allow convenient selection and modification of graphic items.

Isolation Layers

Major subsystems of the IUE will be isolated by standard interface protocols to help insulate the system development from rapidly changing component designs. For example, the Programmer's Interface Kernel (PIK) might serve as an interface to image processing libraries.

Algorithm Evaluation Tools

The IUE will provide support for comparison and testing of image understanding algorithms. This support

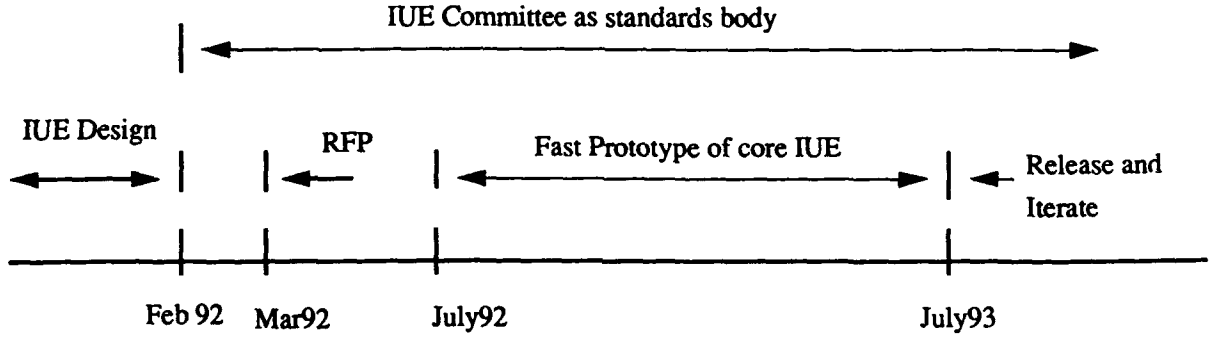


Figure 1: The development schedule for the IUE.

will include database management for test suites of images and other data as well as results of standard algorithms on the same data. Statistical tools will be provided to assist in the determination of classification rates and algorithm reliability.

5 The IUE Abstraction Hierarchy

Object Abstraction for the IUE lays out the trunk of the class hierarchy and proposes software development guidelines. Its aim is to smooth the way for independently developed IUE components to work together. The IUE class hierarchy is organized around a single root class (*Object*), a metaclass (*Class*), and a core set of classical mathematics, physics, and information processing related classes. Guidelines for extending the class hierarchy are also included.

The IUE will be a large and extensible system jointly developed at numerous sites throughout the country. Bringing order to the IUE build requires coordination well beyond generic good intentions of "object-oriented programming" or "OO design". The IUE will ultimately rest on a foundation of one specific class hierarchy and one specific set of design principles. That foundation (not the somewhat vague property of being OO) must support the built-in IUE capabilities and accommodate new additions. A good foundation will keep overall building costs down and improve chances for long-term satisfaction.

Object Abstraction aims to help developers provide consistent capabilities and names for classes and methods. It also aims to ensure that capabilities are comprehensive and logically arranged with well-defined paths to obtaining maximal efficiency. The overall effect is to allow each new development to be added to the class hierarchy at the logically correct point (rather than, for example, making a new incompatible class hierarchy or work with none at all).

Object Abstraction is essentially one *Object* class, one *Class* class, a *Collection* class hierarchy, and design/programming guidelines for the IUE build. These foundation classes define an extensive set of methods for interacting with environment-level tools, for tailoring specialized classes via parametric hooks, and for consistently accessing common mathematics and physics operations. Figure 2 outlines the class hierarchy. The figure is no more than an outline: the IUE has far more classes than shown, name choices are in flux, specific inheritance paths are changing, and the the IUE uses multiple inheritance (albeit quite selectively).

The notation used in the diagram follows the convention of a object-oriented design tool called Object Management Tool or OMT. The defintion for the icons used in constructing the object hierarchies is shown in figure 3.

5.1 *Object* and *Class*

Together, *Object* and *Class* define the environment-level behavior shared by all objects. All IUE classes inherit from *Object* and are associated with a unique instance of the class *Class*. These classes allow the environment to examine instances and configure operations (especially I/O, copy, and display/editing) based on different types

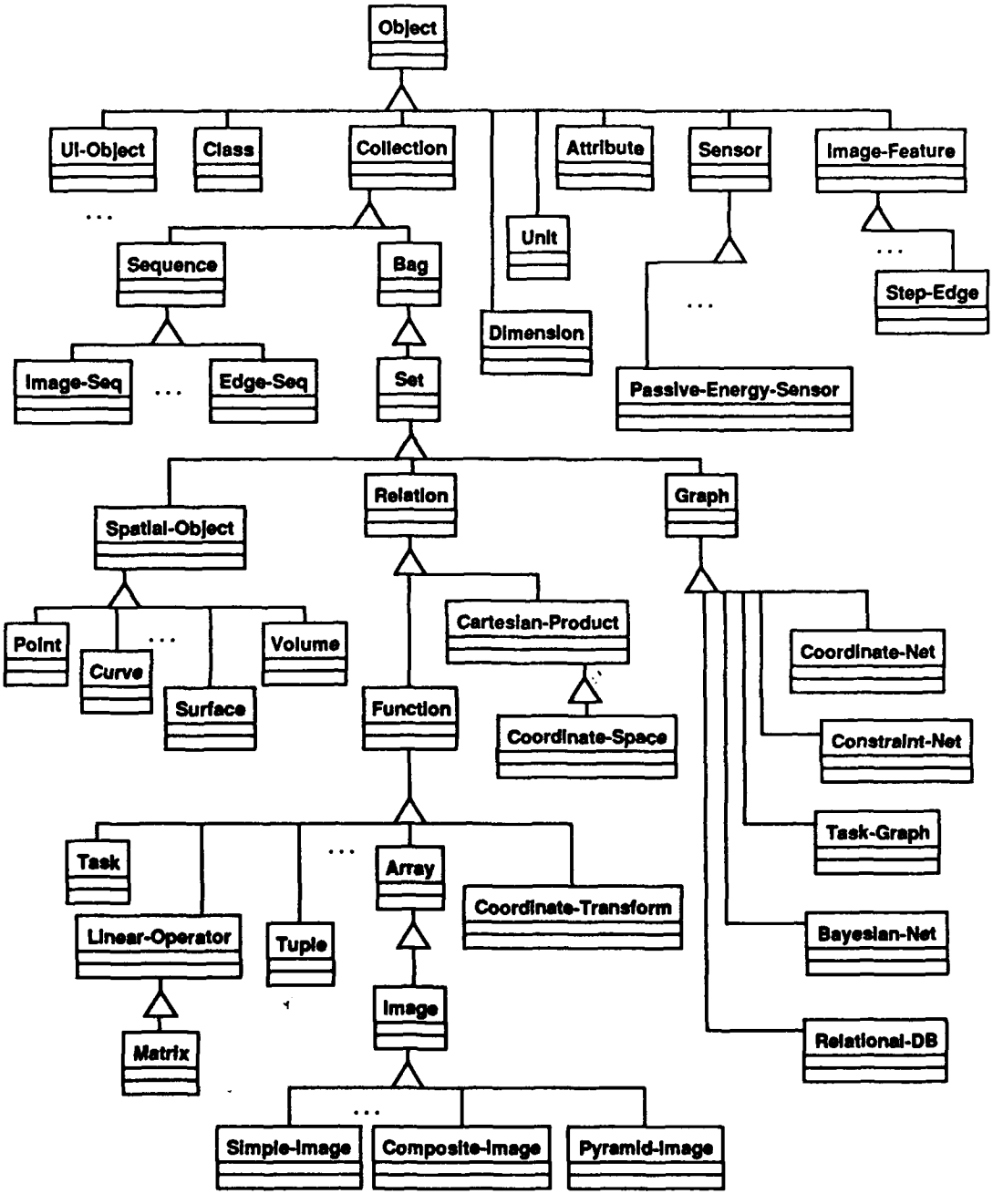


Figure 2: Object hierarchy outline.

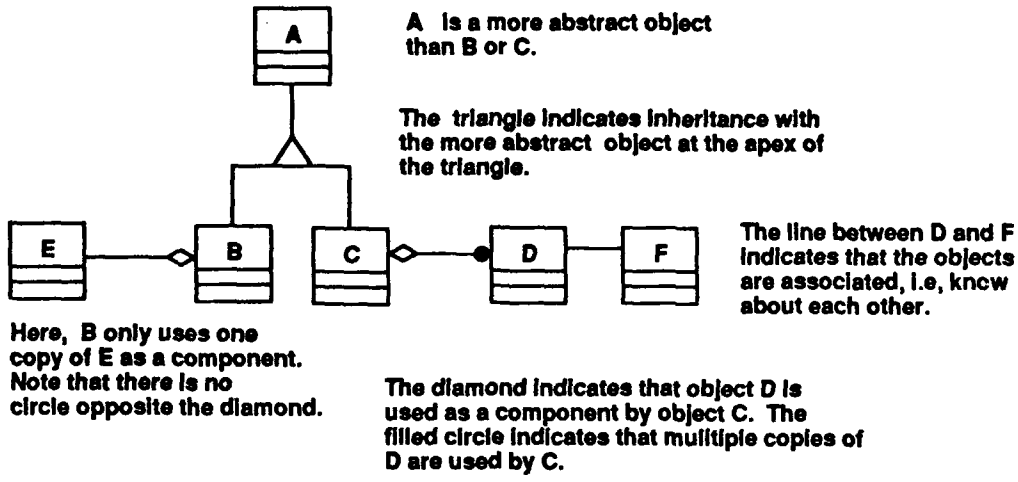


Figure 3: The icons used to construct the object hierarchy diagrams used throughout the text.

and classes. This two-class approach is closely aligned with systems such as CLOS, Smalltalk, and the National Institute of Health C++ Class Library. Applying the approach to C++ requires substantial infrastructure and discipline, but will be valuable in organizing the IUE and providing powerful interactive capabilities.

5.2 Math, physics, and information processing classes

The math, physics, and information processing classes represent such fundamental concepts as images, extracted features, world objects, pure geometry, transformations (including coordinate systems), sensors, sets, sequences, relations, and networks of relations. The *Collection* class is basic to much of the class hierarchy.

Collection is parameterized with functions (including equivalence, insert compatibility, and union compatibility) so that it can cleanly specialize in multiple directions:

- Finite, countably infinite, and uncountable (at least conceptually so) numbers of elements
- *Object*-valued or language-primitive-valued (e.g., int) element types
- Constraints on types or values of elements to be inserted.

Early versions of the IUE design had nearly every class branching of the trunk on Figure 2 inheriting directly from *Object*. The current designs enforce much greater uniformity among important mathematics and physics classes by moving them to more meaningful positions further down the class hierarchy.

5.3 Development guidelines

The development guidelines of Object Abstraction include:

- A dictionary of translating between terminology for C++ , CommonLisp, and standard OOP.
- Naming conventions for methods — these specify such characteristics as return type, inlining, image boundary handling, immutable versions, etc.
- Notions of abstract type and implementation hierarchies embedded in the class hierarchy — this separates method definitions from slot definitions so that highly constrained objects far down in an inheritance hierarchy can have the union of all supertype methods names without carrying around extra slots defined in many unrelated implementations of supertypes.
- Use of source preprocessing tools to aid C++ development.
- Use of persistent object techniques.
- Views of objects — views objects wrap around other instances to change the apparent set of methods or interface (e.g., a view is a low-cost and consistent way of creating a vector-valued image from a sequence of images and visa versa).

The guidelines aim at producing efficient class hierarchies free of semantic conflicts on methods or slots (i.e., potential conflicts due to multiple inheritance are not left to chance).

6 The User Interface

6.1 Scope

The user interface supports many different functions: algorithm and system development, visualizing and monitoring execution, interacting with IUE objects, finding results and information, generating animations and video, and many others. In our design we have studied several different IUEs to determine the critical functionality that we want to incorporate into the DARPA IUE. We have also stressed creating an interface which will be supported by ongoing and future developments in the software world at large. To achieve this, we have organized the interface into a relatively small set of objects which can be built on top of existing interface packages and interface construction toolkits.

6.2 Conceptual Description

The Interface is described in terms of three levels (Figure 4). The **Graphics Level** is the underlying "machine independent" package for basic display and graphic operations and telling the screen what to do. The **Interface Kit Level** consists of existing packages for the creation and rapid prototyping of user interfaces and related tools on top of graphics level software. This also includes tools found in the selected software development environment such as editors and debuggers. The **Image Understanding Environment User Interface (IUEUI) Level** consists of the objects in the user interface. This includes such things as object displays, plotting displays, several types of browsers, and structures for describing the interface context. The IUE interface objects are organized into three basic classes (See Figure 5). The first class consists of displays and browsers. These are the basic tools for viewing an object and inspecting it's symbolic attributes and relations (There are many commonalities between these objects that suggest a meaningful and general IUE interface object). The major portion of what a user does with the interface will be based upon these objects. The second class are the objects commonly used in the supporting user interface mechanisms provided by the graphics and toolkit level (menus, widgets, icons, etc.) but with simplified commands so they can be manipulated directly by IUE users. The third class are support objects for such things as describing the current interface context, the mappings from an spatial object onto a display window, links between IUE interface objects, animation files, and several other things. Many of these are not necessarily full-blown objects, but common data structures.

6.2.1 Object Display and Browsers

Object Displays: These are for viewing objects which have coordinate systems associated with them and mapping them onto a 2D display. It includes such things as images, curves, regions, object models, surfaces, vector fields, etc. They support several types of operations for controlling the mapping of an object to be viewed in a window and for interacting with a displayed object. There are several subclasses of displays that will appear to the user to occur in the same type of window.

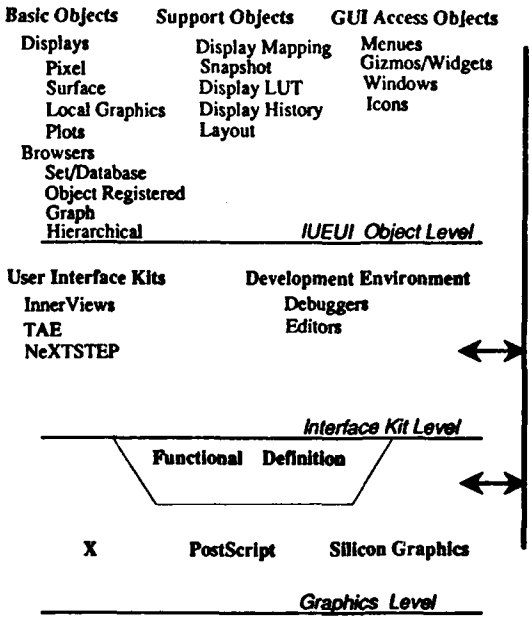


Figure 4: A conceptual portrayal of the various levels of the IUE user interface.

They are primarily distinguished by the types of methods they understand and all inherit a large number of similar methods from the general display class. For example, the overlay method means something different in the context of a surface display than in the context of an image display. The **pixel** display class is for viewing images and image registered features. The **local graphics** display class displays objects by mapping their values onto graphic objects such as lines and cubes. Examples are displaying vector fields and edges. The **surface** display class if for displaying objects that get mapped onto mesh or rendered surfaces. There are several different types of **plot display**: 1D, 2D, 3D graphs, histograms, scatter grams, perspective views of functions and tables.

Browsers: These are used for actions such as queries over set of objects, determining and inspecting relationships between objects, process monitoring, and inspecting values in an object. There are 2 different types of browsers: **Field-Browsers** and **Graph-Browsers**.

Field Browsers consist of a regular array of fields. Fields can be filled with text, icons, colors, colored text, text in particular fonts. Fields can have actions associated with them when they are selected or a user changes the values in them. We distinguish between four types of Field browsers which inherit from the general Field browser class:

- **Set/Database Browser**: This is presented as an array of fields. Each row of fields corresponds to selected attributes of a particular object and each column corresponds to common attributes over the

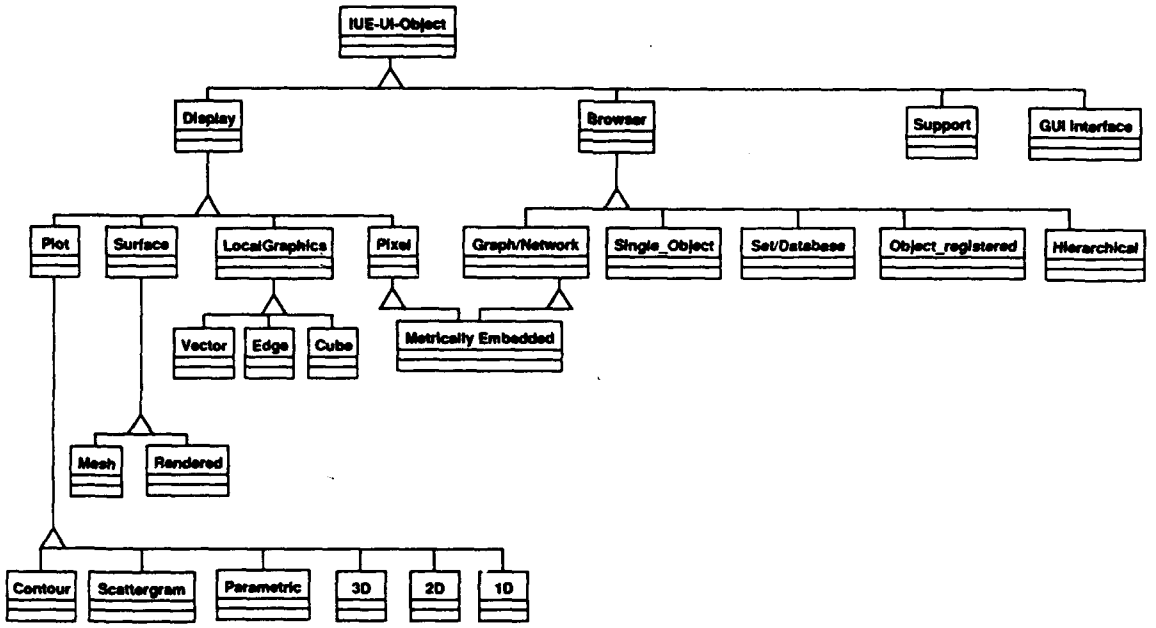


Figure 5: IUEUI Object Hierarchy.

set (or database) of objects. An example would be browsing the database which describes the current active object in the IUE to find the most recently created image from some operations.

- **Single Object Browser**: each row corresponds to the value of an attribute for an object. This is used for inspecting a single object.
- **Hierarchical Browser**: Useful for text based inspection of graph structures and trees. When an item is selected, the related items (along some relational dimension) are displayed in the next column (something like the directory browser on the NeXT machine).
- **Object-Registered Browser**: this contains values extracted from a spatial object, such as the intensity values in some square neighborhood of an image. Depending on the dimensionality of the object (or relationships between component objects), this can be presented as a 1D array, a 2D Array, or multiple 2D arrays and describes curves, images, image sequences, pyramids. There are restrictions on whether it is possible to interactive change values in the fields of an array browser. It should be possible to apply operations directly to the values in the array browser to see the effect of an operation in a restricted neighborhood of an object.

Graph Browsers: These are for the display of graphs and networks, generally representing an object as a node and links to describe relations to other objects. Nodes are similar to fields in field browsers and can be filled with text, icons, colors, colored text, text in particular fonts. Nodes can also have actions associated with them when they are selected or a user changes the values in them. Links can also be colored and selected. A typical use would be for the display of a constraint network.

An important type of graph and graph browser is a **metrically embedded graph** wherein the nodes are restricted to occur at positions with respect to a coordinate system. This type of graph inherits properties from both the Graph Browser and a general spatial object which can be viewed in a display window. An example would be an image registered network which describes potential links between extracted features for displaying grouping operations. An important attribute of metrically embedded graphs is that they can be viewed as an object display for operations such as zooming and having access to the underlying context in an image.

6.2.2 Simplified access to underlying GUI objects

Gizmos and Widgets: The IUE will provide simplified, interactive access to the interface objects found in GUI Kits. Such things as sliders, knobs, buttons, text input/output fields, menu creation and personalization. This will involve commands for creating gizmos and widgets, for positioning and scaling them, for attaching them to parameters, for reading and writing to them. An example would be creating a slider and then getting values for an interactive thresholding operations from it.

Menus The IUE will provide simplified interactive access to menus in the GUI kits. This involves being able

to extend menus, create pop-up menus, associate actions with menu items. A critical design task is deciding what goes into system level menus and how they are organized

Icons The IUE will provide simplified interactive access to icons in the GUI kits.

6.3 Support Objects and Common Data Structures

There are also several objects that are used and manipulated as part of the interface. Some of these are:

- **Display-Look-Up-Table**: A generalization of a color look up table that describes how to map object values onto screen values. It can also include functions.
- **Object-Display-Mapping**: A structure which describes the mapping from an object onto a display. This includes both the position and values of how the object is displayed and a reference to a particular Display Look Up Table.
- **Object-Browsing-Mapping**: A structure which describes the mapping from an object or database onto a browser
- **Object Display Links**: A structure which describes the concatenation of a display or browsing operation between IUE interface objects. Thus a link between display windows w1 and w2 with an associated zoom and pan would display an object in w1 with w1's object display mapping and then display the same object in w2 by concatenating onto the object display mapping for w1, the specified zoom and pan operation.
- **Interface layout**: A structure which describes the object instances in a particular instantiation of the interface. Users may prefer different interfaces (arrangement and instantiation of the basic IUEUI objects) depending on the task or level of sophistication.
- **Display Context**: A structure which describes current context for a display. Such things as the current window, the current object, the current object display mapping, the current display command, the current mouse-selected object position and value, and others. Display operations can use defaults based upon these.
- **Browse Context**: A similar structure for browsing operations. Such things as the current browser, the current data base, the query history, and others.
- **Display Snapshot**: What is produced when the current display is written to a file. It is just what appears on the screen and not the actual objects
- **Animation File**: A sequence of display snapshots

The interface has an interactive command language for interacting with objects and sending messages to displays and browsers. These commands can also occur in code for creating scripts. The interface supports **Interactive Command Buffers** which look like WYSIWYG text editors. Textual outputs can be written to the Interactive Command Buffer. It has a vertical scroll bar for accessing previously written commands and allows operations like cutting, pasting, etc. Another interface component is the **Tool Box**. There are probably hundreds of nice interactive controls for displays and visualization that IU researchers are familiar with, such as interactively manipulating the object-value to screen-intensity function by interactively shaping the function; selecting color look-up tables; modifying color look-up tables; interactively building display commands using templates or command browsers; floating tool palette of interactive drawing tools; etc. The Display Tool Box is a menu of such tools, organized into functional groupings of interactive tools for manipulating the current display. From a functional point of view, it allows potentially redundant access to the display methods without using the interactive command buffer. It somewhat like the system control menu on the Macintosh and the system preference menu on the NeXT machine. Users can also select particular interaction tools and have them occur as floating palette (in cases where the user want to interact with multiple tools from different sets of tools at the same time).

There are several methods for the general display object (and related ones for browsers). They are organized into different groups:

Methods for Manipulating the Current Object display position mapping: This includes operations such as panning, zooming, perspective views, and warping. These are methods that control how positions in the specified object(s) get mapped onto a display window

Methods for Manipulating the Current Object value mapping: This includes operations such as overlays, mapping onto different color bands, transparency, and others. These are methods that control how values in the specified object(s) get mapped onto screen attributes such as color and intensity

Methods for setting the current display mapping table: how to configure planes in the screen buffer for the display of color images; how many panes to use for overlays; particular functions and conditions to apply to object values prior to di splay

Methods for Screen Attributes: These involve controlling attributes of the window the display is mapped onto and includes such things as position, size, attributes of the title bar, event handling for the mouse

Methods for Links: Linking display transformations in different windows. Operations include creating links and associating position and value mappings with the links

Methods for Interaction: These involve interaction and manipulation of displayed object(s) in the display. Operations include recovering object position and value from a mouse click, applying functions to selected objects, applying functions using selected information.

Methods for History: Methods to coordinate displays overtime, such as cycling through an image sequence, playing an animation of displays

Methods for Graphics: These involve accessing display registered graphics packages for drawing lines, text, and other things. These occurs in four different modes: 1) relative to the window of the display; 2) Relative to the entire screen; 3) the specified object or coordinate system; or 4) for instantiating IUE objects corresponding to the graphic displays

Methods for printing, writing to file, animation

6.4 Applications

One aspect of the interface design has been to compile examples for typical or interesting interactions that correspond to task a user may want to perform. Examples are such things as different ways of displaying an image; establishing processing links between multiple displays and browsers; interactively applying a function to a local neighborhood of a displayed IUE object; display of a vector field and 3D surface orientation; finding a previously generated result and displaying it; viewing a surface display of image intensity in the neighborhood of an occluding edge over a sequence of images; and several others. We expect this to expand to several hundred examples by the time the specification is completed.

For example, the following prototype commands;

```
[p w1 image]
[i w1 :1 [s w2 image :location image.x image.y
         :distance 200
         :theta1 30
         :theta2 0
         :clear t]]
```

would display an image in window 1 and then allow the user to interactively select locations and values from the image by mouse clicks in the window. When the user hits the number 1 terminal key, a surface display of the mouse-selected area is generated in the display in window 2.

For general applications, particular emphasis is being placed on 1) allowing the user to configure the interface from the underlying objects and 2) also having default interface layouts and scripts so applications can be used by non-specialists.

6.5 Extensions

There are several extensions we intend to investigate when the initial specification is completed:

- Compatibility with different visualization and plotting packages
- Incorporating Hypermedia and Cooperative Work tools for communication between IU researchers and on-line tutorials
- Compatibility with diverse interaction devices (Currently we are assuming only a multi-button mouse and a keyboard). Future devices interfaces will almost certainly involve gesture and voice

7 Tasks and Tools

7.1 Scope

Tasks are envisioned as the object-oriented mechanisms that are used to represent large-grain IU processes; i.e. the Task class is used to implement *IU algorithms* within the object hierarchy of the IUE. By *large grain processes* we mean those processes that operate over a domain consisting of a significant number of pixels or other IU objects. Thus a function that applies a convolution kernel to an image would be considered to be an IUE Task, but a function that scales the value of an individual pixel would not.

The concept of a Task as an object rather than as a method associated with the traditional IU objects such as images or lines is not necessarily obvious. However there are several significant aspects of these processes that lead us to this model of a Task as an IU object.

Large grain IU operators typically have **complex parameter structures**. A significant portion of the time spent in developing IU applications is spent in the exploration of the search space defined by the parameters of the operators. For example, a common type of question that needs to be answered by IU researcher is: "What is the most effective Laplacian radius when performing a Zero Crossing segmentation on aerial images of cities?". A great deal of time and effort is spent in determining appropriate values for parameters of a particular operator in a particular domain. Once these parameter values have been determined, it becomes natural to think of the parameterized operator as a new entity that is different from the unspecialized generic operator. This view leads to the concept of an operator as a Task object that can use inheritance and specialization to represent these parameter structures.

IU research also involves a very large amount of processing and data generation; in this type of environment it becomes important to be able to examine the **processing history**. The Task class readily supports the maintenance of a processing history through the explicit representation of Task parameters. A Task instance can exist in one of three states depending on the specification of the input and output parameters for the Task: it may either be partially specified, fully specified, or completed. In this way, Task instances describe both the complete input and output specification and the processing status of Tasks. The Task objects thus provide a complete description of the large granularity image understanding processing that has occurred in a user environment.

Another aspect of large grain IU processes is that they are used by the IU researcher as a **set of tools** within an experimental toolbox. Researchers require a flexible mechanism for control and data chaining that allows them to construct experiments that combine individual Tasks into more complex algorithms. The Task class hierarchy provides this mechanism through the *CompoundTask* and *DataflowGraph* object classes. With these classes, the user may chain together individual Task objects either through programs or through the use of a interactive graphical interface.

DataflowGraphs allow the user to specify data path-

ways between Task objects. The Tasks in a DataflowGraph can have some subset of their input parameters specified dynamically; the input values of the dynamic parameters are specified by the values of output parameters of other Task objects. Whenever values have been specified for all required input parameters, the Task object is executed. Other DataflowGraph objects, such as DataflowConditional nodes and DataGenerator nodes, provide the control constructs that make the DataflowGraph an effective programming tool. A DataflowGraph may be constructed either in C, LISP, or at the interface level, to form these complex processes. In C or Lisp, this complex Task control can be also implemented through a message passing paradigm in which Task instances are parameterized and controlled through messages (generic function calls) from a controlling program.

Programming modularity is provided through the CompoundTask class. Whether the process is specified through a DataflowGraph or through a message-passing program, the entire process-specification may be abstracted into a single CompoundTask entity. These CompoundTasks may then be appropriately parameterized and used as Task objects in the interface.

Another specialized aspect of IU algorithms is that they typically have **high computational demands** and can benefit greatly from multiprocessor configurations or special purpose hardware. Because the Task model separates the I/O specification from the process specification, the execution of a Task object can become independent of the host language and processor. In this way, it becomes possible for the user to work within a single, unified environment, but at the same time have access to a wide variety of processing options.

7.2 Conceptual Description

The significant classes of the task object hierarchy are Task, ProcessSpecification, TaskGroup, and DataflowGraph.

Task A *Task* is the object level representation of an large grain IU algorithm. It explicitly contains a complete description of the I/O parameters for the process corresponding to the algorithm. It also contains a reference to the executable code for the algorithm in the form of a ProcessSpecification object. Because this ProcessSpecification is separate from the Task itself, the Task model supports a wide variety of processing environments.

ProcessSpecification A *ProcessSpecification* object is designed to be a point of access to the executable code which implements the algorithm for a Task. This specification allows the actual code to be specified either in the native language (Lisp or C) for execution within the current process, in the non-native language for execution within the current process, or in either language for execution within an external process that may (potentially) run on a separate processor.

TaskGroup A *TaskGroup* is the object level representation of a collection of related task objects. TaskGroup objects provide an graphically oriented experimental vehicle for the easy exploration of a variety of related algorithms and also provide a representational hierarchy for the Tasks of the IUE. Typically, the set of Tasks specified in a TaskGroup will possess very similar I/O parameter specification, thereby making it easy to switch between the various Tasks.

DataflowGraph A *DataflowGraph* object is the representation of the dataflow between Tasks in a ProcessSpecification. Specifically, it is a graph of DataflowNodes. The graph describes the data dependencies between Tasks. The DataflowNodes themselves describe the exact mapping between the output parameter values of one Task and the input parameter values of another.

7.3 Applications

The *Tasks* that will be supported by the IUE will cover a wide range of algorithms and tools. It will be expected that the set of Tasks that are included with the IUE will expand rapidly as the IUE begins to receive wide use and the research groups using the system begin to contribute their own research tools.

The following sections are suggested as a quick first pass at the set of standard TaskGroups that will be supplied as standard components of the IUE.

Browsers Interactive graphical tools for textual and symbolic examination.

Display Tools for display of image and symbolic data.

Editing Tools for the creation and manipulation of image and symbolic data.

ImageProcessing Tools that map image data to image data.

ImageSegmentation Tools that map image data to symbolic data.

PerceptualOrganization Grouping tools to map symbolic data to symbolic data.

GeometricFitting Tools that fit geometric entities to symbolic data.

ObjectMatching Tools mapping object descriptions to symbolic data.

ModelConstruction Tools for creation and manipulation of object descriptions.

7.4 Extensions

There are a large variety of extensions that can be made to the basic Task hierarchy. One such class of extensions involves control of complex processing models. The Task hierarchy should be extended to provide explicit interactive control for multithread processes and for massively parallel processors. The hierarchy should also support shared memory models and the concept of *server* tasks that run continually and provide data on demand.

A second class of enhancements involves software and algorithm development tools. The Task hierarchy should support complex algorithm development tools that provide:

History mechanisms Integrated development tools that allow the user to select sequences of operations from the processing history and then rerun the Tasks with new parameterizations.

Data management Database tools that allow the user to easily store and retrieve experimental data based on the processing history and vice versa.

Debuggers Debugging tools that allow single stepping, error trapping, watchpoints, etc. over the Dataflow-Graphs as well as the ability to step into the Task processes that are spawned from the Dataflow-Graph.

8 Image Classes

Images are the most basic and fundamental concept in computer vision. They provide the sensory link between events in the world and the internal representations of these events within the system. In many cases, an image can be simply viewed as a two- or three-dimensional array, as in the case of static grey-scale and RGB color images. This section preserves this notion of simplicity of representation as far as possible, while also accounting for the wide variety of image structures found in the vision literature.

In the image object hierarchy shown in Figure 6, images fall into one of two subclasses: simple or composite. A simple-image is a specialized form of an N-dimensional array ($N = 2, 3, 4, \ldots$). A great variety of common forms fall into this class: color and grey-scale images, range images, CAT images, image sequences, stereo pairs, etc. The composite-image class supports images that do not fit naturally into a single N-dimensional framework, such as image pyramids, or mosaics composed of partially overlapping images. Composite-images are essentially sets of images, with additional slots and methods to support specialized semantics. Since composite-images are themselves made up of simple-images and composite-images, ultimately all pixel data resides in simple-images.

In addition to the inherent structure of images, two other considerations (efficiency and redundancy) have influenced the design of the image object hierarchy. The object oriented approach makes it imperative that multiple image objects be able to share pixel data. Otherwise, it would be necessary to redundantly represent parts of images in order to apply methods defined for these parts. To give an example, consider the red plane of an RGB image. It has the same form as a grey-scale image, a 2-dimensional array of values. To apply a grey-scale image method to the red plane data a grey-scale image object must be created. To avoid redundant copies of the data, this new object should point to, and hence share, the same underlying data.

Sharing is a desirable property of arrays in general, not just images. This, in addition to efficiency considerations, is why arrays are included here. All the mechanisms that support data sharing between multiple objects are provided by the subclass shared-array, and are thus inherited by simple-images. Shared-array objects, in turn, use raw-data objects to implement sharing; two or more shared-arrays share data by pointing to a common raw-data object.

There are three distinct interfaces associated with simple-images. Simple-images inherit the image interface from the root image class. They inherit an array interface from the array class. Finally, a very low level but very efficient interface is provided through the raw-data objects themselves.

From the root image class, simple-images inherit slots and methods associated with the sensor used to produce the image. Also from the root, simple-images inherit pixel semantics, including pixel access. Since most, though not all, image sensors utilize a two dimensional sensor grid, and since pixels are commonly associated with individual cells in this grid, most images are two dimensional. To illustrate, get-pixel with arguments (x, y) returns a three place vector for an RGB image.

From the shared-array class simple-images inherit methods for multi-dimensional array access. They also inherit the map-slice and map-copy methods. These methods create new images which map onto subspaces of the first. The map-slice method does this without copying the data and hence the new objects share elements with the original. To illustrate, an RGB image is a three dimensional shared-array. Array access with three arguments, (x, y, c), may be used to retrieve values from the image. The map-slice facility may be used to create a new simple-image containing only the data associated with a single color plane. As a second illustration, map-slice may be used to generate a temporal slice through a motion sequence. The map-slice facility is quite general, and can be used to create a new image containing elements in any subrange of the original image. It also supports subsampling.

The third interface is directly through the underlying raw-data object. A raw-data object is a first class object; it is not an object the casual user should be concerned with. It is provided for the benefit of sophisticated users who need direct access to data. A raw-data object maintains a pointer to a block of memory, or perhaps a set of blocks. It keeps a list of all other objects which point to this memory. It also maintains information about the layout of the data within the block. Layouts for many of the standard types of shared-arrays will be published in appendices to the IUE Programmers Guide. Using this information, a systems programmer can write code that performs its own pointer arithmetic to directly access the data. Shared-arrays interpret the data in a raw-data block using a set of index vectors. Each shared-array has its own set of index vectors, and uses them to map between multi-dimensional queries and a single sequential index into the raw-data block.

Composite-images, unlike simple-images, do not have a well defined multi-dimensional form. A composite-image is essentially just an ordered set of images. The class is important because it provides flexibility. For example, there is no efficient way to represent an image pyramid within an N-dimensional array. However, since composite-images are not shared-arrays, the slice methods are not available.

The methods defined for the image object class in-

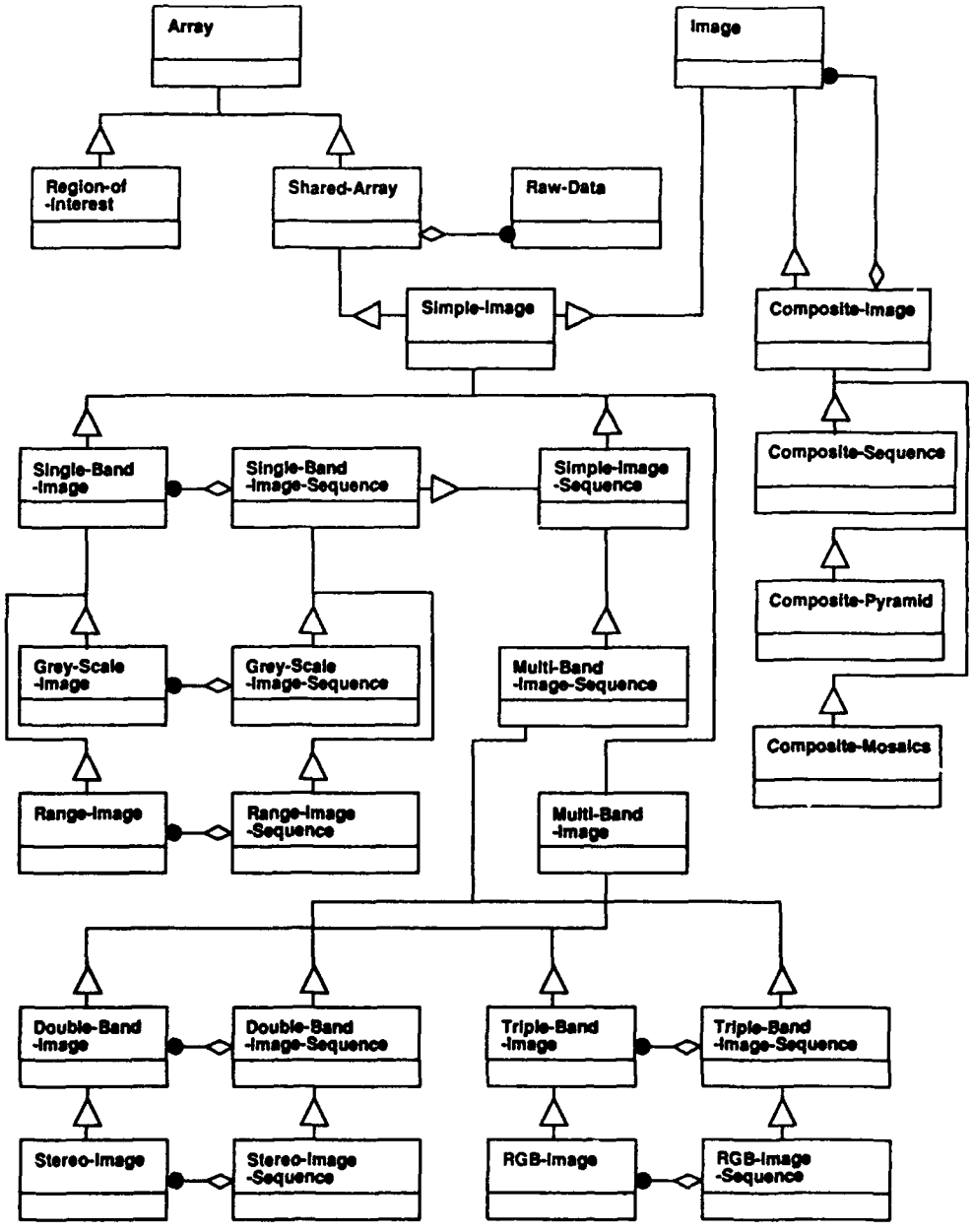


Figure 6: Image object hierarchy.

clude those required for creating, deleting, modifying, accessing, querying, and displaying instances of the object class. The image object class interacts with the other objects in the IUE system through the methods which create instances of these objects, e.g. regions, edges, lines, and various other geometric and non- geometric features. An open question before the IUE committee is, among the perhaps thousands of candidate algorithms in the vision literature, which will become the basic methods of the core IUE system. One possible solution is to include in the core those basic techniques which are common to a wide variety of algorithms, such as convolution, histogram creation and manipulation, filtering and enhancement, etc. The more specialized methods, such as the Ohlander-Price region segmentation algorithm or the Burns line extraction algorithm, would be included in libraries of methods available outside the core system.

9 Image Intensity Features

9.1 Edge Events

9.1.1 Scope

Image features play a central role in image understanding research. The reliable detection of image intensity boundaries has been the goal of many image understanding research projects. The approach taken in the IUE is to provide a basic framework for representing intensity events and to supply the most widely used image feature segmentation algorithms.

A great deal of initial discussion during the formulation of the IUE centered on approaches to the representation of an edgel, since there is a diverse spectrum of opinion within the IU community as to the proper description of an intensity event.

The description of step edge elements, and other boundary events, in the IUE is in terms of the geometry of the intensity surface in the neighborhood of the the event. The need of specific algorithms for additional attributes is met by providing a flexible attribute mechanism so that special properties can be attached to the basic geometry description. A major contribution of the IUE committee is to define a standard set of attribute names so that other algorithms associated with image events can rely on a consistent definition of attributes.

9.1.2 The Edgel

The class hierarchy, shown in figure 7, illustrates a definition for a step edge. The description includes the position and orientation of the edge as well as attributes of the intensity cross section at the edge point. Examples such parameters are the average intensity, the intensity change across the edge and the slope of intensity along the boundary. The definition of edge filtering kernel is provided by the IUE filter object hierarchy. The edgel object lists the kernel type as an optional attribute.

It is often necessary to provide statistics of these properties in the neighborhood around the edge position. A convenient approach is to define a covariance matrix which expresses the variance and correlation between image features. These statistics can be used later in a decision procedure for classifying the image events.

9.1.3 Spatial Index

It is usually necessary to group edgels into a connected chain according to a geometric relation such as collinearity. This grouping process is facilitated by a spatial indexing mechanism. In the case of collinearity, an appropriate spatial index is the Hough array. If the edgels are sorted into the array, then grouping on collinearity becomes quite efficient. The design allows for other types of grouping, such as convexity, with the definition of an associated spatial index.

9.1.4 Edge Sequences

The grouping of edgels has as a goal the formation of a one dimensional chain which terminates at junctions or end points. Junctions are formed by the intersection of two or more edge chains and endpoints are where a boundary simply terminates. These sequences may have missing edge segments where there was no strong evidence for a boundary, but the grouping mechanism introduced a link for the purposes of continuity.

9.1.5 Extensions

There are many variations on the basic representation described here. In the case of range data, it is desirable to provide a second order surface description of the neighborhood around the edge event. The average surface normal, and curvatures on each side of the boundary can be provided. Similarly, the details of a peak type of edge event will require descriptions of curvature in the edgel neighborhood. Also the position and orientation parameters should include provision for subpixel interpolation.

9.2 Regions

9.2.1 Scope

A **Region** is an area of an image (or surface) produced by operations such as histogram partitioning, region-growing, model surface projection, and many others. Regions are two dimensional bounded surfaces and have many common methods with the general surface object. For example, the composite regions produced by image segmentation are a multiply connected face with a set of 1-cycles enclosing pixel areas in the image plane. Regions have been specialized since they are a basic type of feature extracted from images and projected from surfaces and volumes. For example, composite regions have attributes and methods to access properties of adjacent regions sharing a common boundary for use in grouping operations. Non-geometric attributes can be added to regions using the mechanisms for associating attributes and values with the general IUE object. The mapping composition method is used to register a region with an image and to access the corresponding image values. This is used for computing attributes such as the average or variance of image features in the area defined by a region.

Beside being a basic feature type, the IUE region class is interesting for several reasons:

- Regions are very similar to several other IUE classes such as curves, surfaces, and geometric features.

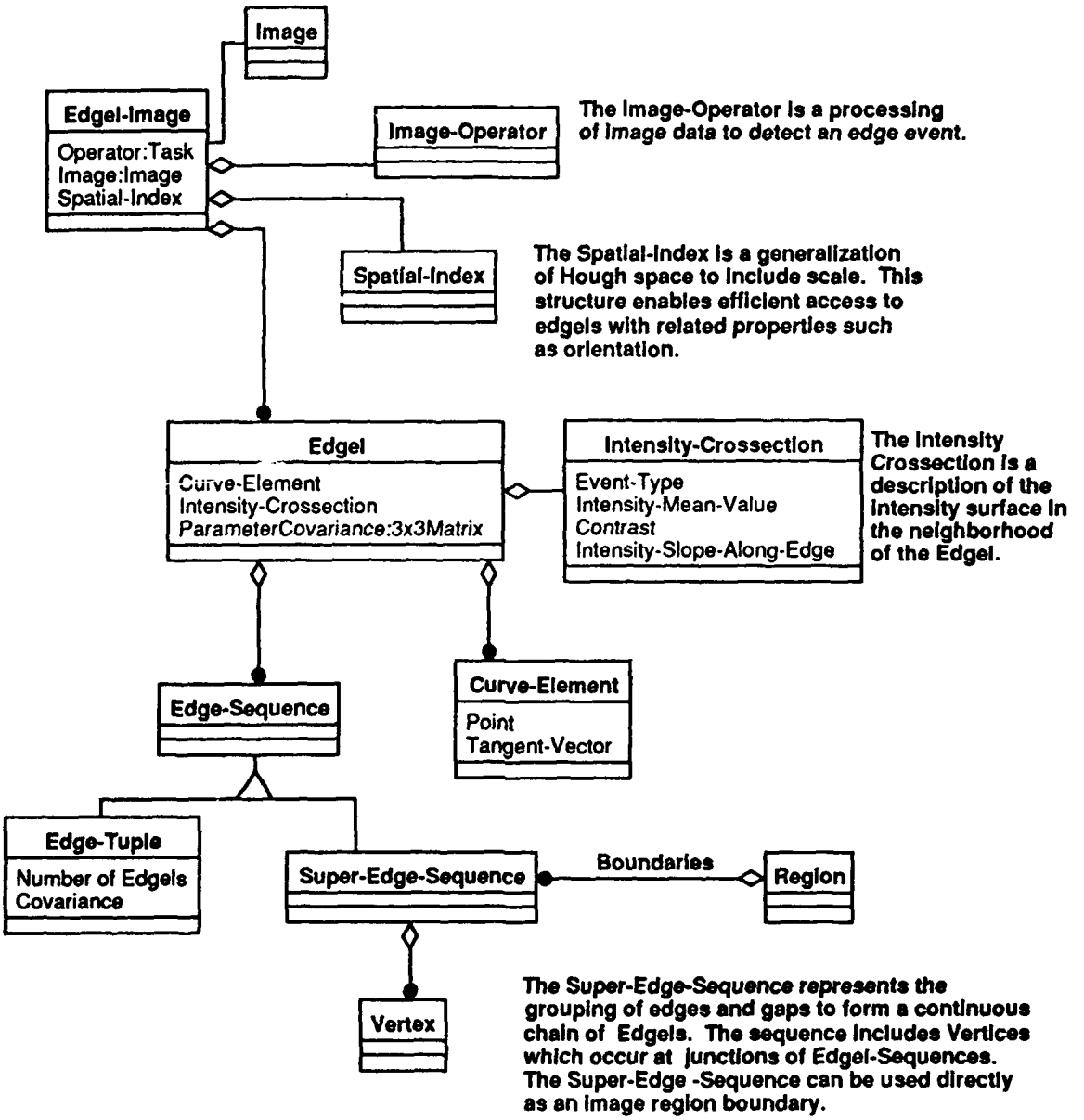


Figure 7: A partial hierachy for edge events.

This has suggested underlying abstractions common to all these objects.

- Regions are useful for spatial queries, especially for grouping operations
- Many image operations can be specified as though they occur with respect to a region which is then iteratively positioned and registered with an image. A typical example is a mask or extracting some portion of one image as a region and then matching it with another image.

9.2.2 Conceptual Description

The general region object is described by a closed outer boundary, a set of interior boundaries, a coordinate system, and a set of points contained in the region. Not all of these are necessarily specified for a particular type of region (In fact a region can just be described as a point-set of image positions it occupies). Some regions do not require an explicit boundary (BitMask Regions). Some do not require an explicit point-set for points contained in the region (Analytic Regions). Some do not require an explicit reference to a coordinate system because the coordinate system will default to that of the image from which the regions were extracted. The general region object has attributes for Area, Number-of-Holes, Minimum-Bounding-Rectangle, Centroid, Scatter-Matrix-of-Pixel-Positions, Compactness. These also may or may not be specified in a region instance. This list is by no means exhaustive. The IUE will define a large number of other region attributes such as intensity co-occurrence matrices, texture primitives and color classification parameters.

Region boundaries are instances of curve objects. Different subclasses of regions are distinguished by the types of closed curves which describe the boundary: a pixel chain; a parametric curve; a connected sequence of line-segments; a piece-wise polynomial. Many of the instantiation methods for regions involve filling a closed curve or fitting a region to a set of points or a set of curves. Boundary type also effects determining whether a point is inside a region.

Analytic, composite boundary, and discrete regions correspond to the similar classes found in the curve class. BitMap Regions are binary arrays with values corresponding to which pixels are occupied in the minimum bounding rectangle of the region. They also have a specified position with respect to an image. BitMap Regions can be used for rapid determination of intersections, and searching operations.

The methods associated associated with regions fall into several basic types:

- Creation methods: These are for creating instances of region objects. Many of these involve creating regions of one subclass from another subclass. The different analytic classes can be instantiating by fitting them to a discrete region. A discrete region can be instantiated by sampling an analytic region at some resolution. A polygon can be instantiated from a sequence of points or from a convex hull fitting routine applied to a point set. Composite regions can be instantiated from a connected components image.
- Combination methods: These are for creating regions from a set of regions using operations such as intersection, union, difference, merging (combining adjacent regions into a composite region)
- Mapping methods: The are general methods based upon the semantics of mathematical mappings for operations such as composition, embedding, and iteration. For example, a region can be composed with an image such that positions in the region can be used to access values in the image. Or a region can be iteratively moved with respect to an image to perform operation with respect to a swept area.

A partial hierarchy for region geometries is shown in figure 8.

9.3 Extensions

Additional study is required to arrive at a suitable object hierarchy to reflect the variations in region grouping mechanisms and to account for different region elements. It is expected that regions will branch similarly to image object types, e.g. intensity regions, RGB regions. At the time of this writing little attention has been given to texture properties and texture grouping mechanisms. It is expected that the final hierarchy will express the basic texture grouping mechanisms and texture element categories.

10 Geometric Features

The type of IUE objects that are described under the heading of Geometric Features are a subclass of general Spatial Objects and include simple features (e.g. lines) extracted from an image and large or small collections or combinations of these kinds of features. A sample hierarchy is shown in figure 9. Since these are the type of features that play role in the low-level to mid-level processing of image understanding, these features interact with other object classes that provide low-level and mid-level representations. A collection of geometric features is usually derived from an image object and shares properties of an image, but not usually the iconic representation associated with an image object. Image features, such as in the edgel class, are extracted from the image and processed to form the line or curve segments discussed here. Lines and curves are general object classes also discussed in another section and form the basis for the descriptions used here. Other image derived features such as regions are also described elsewhere.

10.1 Basic Classes of Geometric Features

We have divided geometric features into three primary classes of objects: the curve segment (an interval on a curve bounded by two vertices), the curve segment group (grouping of individual curve segments by some criteria), and the geometric object image (the set of such features extracted from an image). The first is the basic element used in the later descriptions, the second is a small collection of such elements, and the third is a large scale collection of both basic and grouped elements.

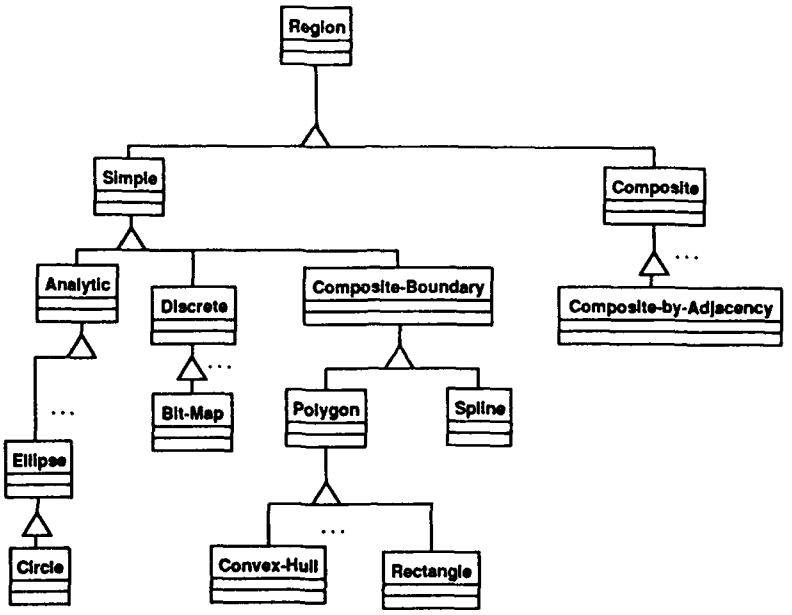


Figure 8: A partial hierarchy for image regions.

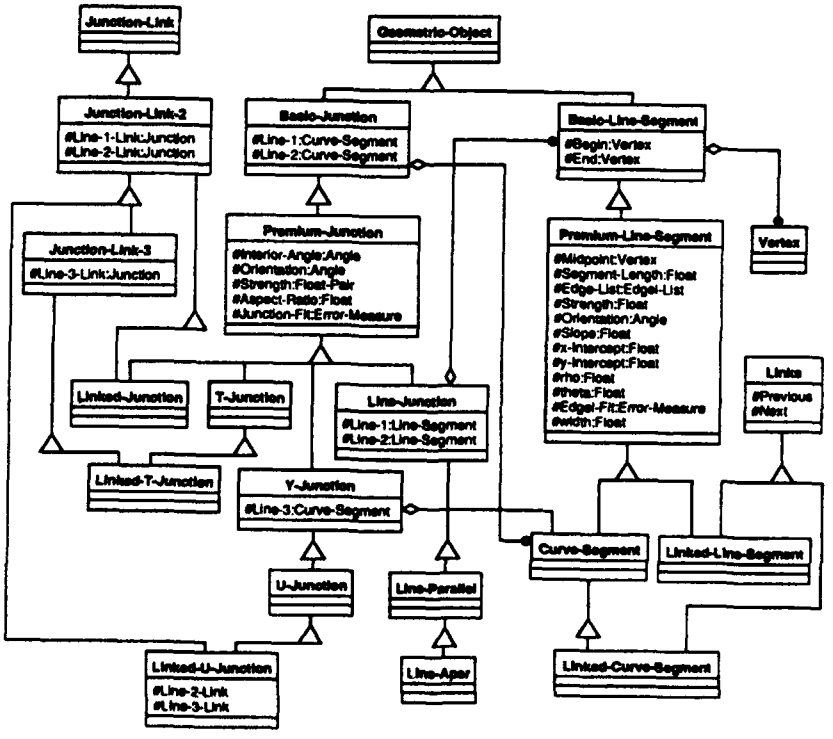


Figure 9: An object hierarchy for geometric features.

The curve segment class, and its more common specialization the line segment class, stores the representation of curves (or lines) extracted from an image. The simplest form of this representation needs only the beginning and ending points of the interval, and a representation of the curve for general curves. In normal usage, other information must be stored because of the difficulty of recomputing without resorting to other information (e.g. integrated edge strength, list of corresponding edge elements) or because of the expense of recomputing frequently used values (e.g. length, orientation). We have thus defined a more complete representation of curve and line segments than will normally be used in order to accommodate these other values and to give names for the value access or value computing functions. A second class associated with the segment class is the linked segment class which connects segments into ordered lists such as would be derived when chains of edge elements are converted into straight line approximations.

The curve segment group class includes small collections of segments that have been grouped according to some grouping function. In general this object class allows for an arbitrary grouping function with an arbitrary number of grouped elements, but the more common segment groups are those with two or three segments such as anti-parallels (two nearby segments with opposite direction), junctions of lines (L-junction, Y-junction, etc.), etc. where the grouping function is implied by the object class. Other more complex perceptual groups contain similar information, but also must contain a description of the perceptual grouping function, which in an extreme case may be a pointer (e.g. a text string) to a large user program with certain arguments specified.

An image is used to store an iconic representation of some scene produced by a sensor. A geometric feature image is a collection of geometric features that correspond to some image, thus it will have many of the properties associated with an image in the IUE. This is treated differently than a perceptual group where the grouping function is based on derivation from some image because it is the common form for referring to the large collection of geometric features generated while processing an image.

10.2 Applications of Geometric Feature Classes

A typical use of these object classes is in the extraction of ribbon-like structures in an image. One possible way for this to proceed is to apply an edge operator to an image followed by a procedure to extract straight line segments (specialized from of the curve segment class) from the set of linked (adjacent) edge elements. This line segment image is then processed to extract a large number of pairs of segments that are anti-parallel and a specified distance apart (according to the ribbon width), which are objects of the apar feature class and are stored in another geometric feature image. These basic (two element) groups are then combined to form a small number of much larger perceptual groups (members of a perceptual group class that stores the members and the grouping function) using a complex process that depends on colinearity, properties of gaps, etc. and these few groups are then used by the program for further processing.

10.3 Expected and Planned Extensions

Since one of the goals of the IUE is extensible descriptions, these must always be allowed. These classes are meant to serve first as a basis for implementing programs and second as a model for adding new object classes when needed. Obvious extensions include the addition of other line junction types if needed for a particular application, other (non-intersecting) small segment groups, and specialized groups such as symmetries. These could all be added by modifying the similar classes that are already defined. The more complex extension to higher dimensions would require adding new classes that inherit from the existing classes and from the higher dimension spatial feature class. Few, if any, slots would be added since the endpoints are already treated as a vertex, which can already be of any dimension.

11 Curves

11.1 Scope

In this section we describe the necessary objects that are used in computer vision literature. We provide a brief description of the objects and the motivation behind our choices. The objects described includes arc representations as well as data structures, such as points and pointsets, from which such arc representations may be computed. A portion of the object hierarchy for curves is shown in figure 10.

11.2 Points and Pointsets

A point is a spatial-object. We classify points as attributed or non-attributed points. A non-attributed point contains only the coordinates. The attributes are required since many algorithms use additional information such as edge direction, edge contrast, graytone surface curvature, etc. An attributed point is a simple point with added attributes. Specifically, an attributed point will have an attribute list that specifies what attributes are stored and a pointer to an area of memory containing attribute values. The various attributes allowed may include (but are not limited to) edge direction, edge contrast, curvature, accuracy/tolerance zone for a point, covariance matrix, transformation matrix etc. Algorithms such as the Hough transform method use edge direction as well as the edge strength. Other recent robust techniques that propagate the uncertainty from the low-level stage need the covariance matrix or the tolerance zone information. The transformation matrix information may be needed to just indicate that the algorithms operating on the data have to use the transformed values while leaving the original coordinate values stored intact.

A pointset is a collection of points that may be ordered or unordered. Points in an ordered-pointset are ordered as they are encountered along an arc-sequence. Ordered pointsets, in general, are the inputs to algorithms that perform arc-segmentation. An ordered pointset is a digital arc. A pointset may be attributed. Each point in

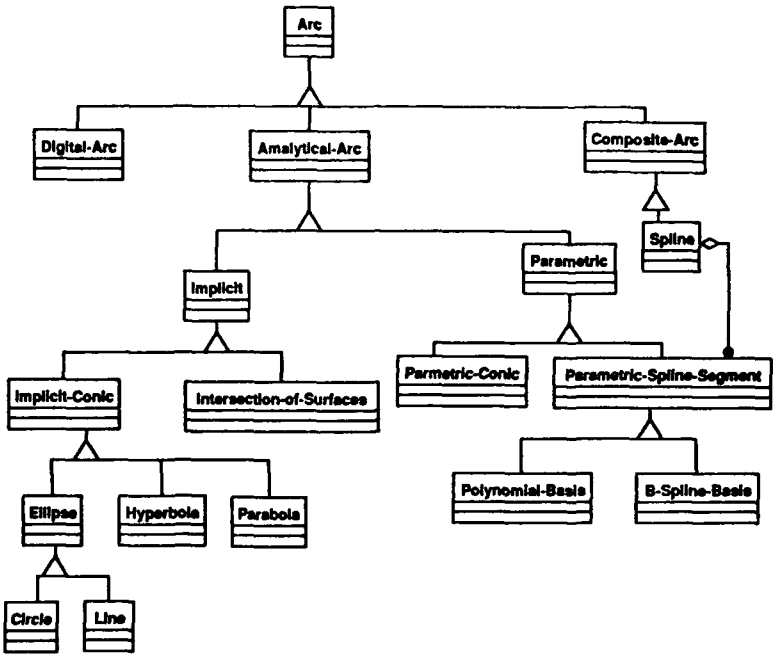


Figure 10: A partial hierarchy for curves.

the attributed pointset may be attributed, or there may be a collection of attribute values that are common to all points in the set.

11.3 Arcs

An arc is a spatial-object. An arc may be a digital-arc (an ordered sequence of points) or an analytical-arc (analytic representation of an arc). The class "arc" inherits all attributes of "IUE-object" and in addition stores coordinates of the starting point and the ending point of the arc. An ordered-pointset is a digital-arc. A digital-arc-chain is a segmentation of the digital-arc into its continuous subpieces. A typical arc-segmentation algorithm would take a digital arc as input and produce a digital arc chain as output. Some algorithms take a digital-arc-chain as input and produce a new digital-arc-chain as output. An example of such an algorithm would be the breakpoint optimization algorithm which operates on a segmented arc and moves the breakpoints to produce an output segmented arc.

Analytical-arc, an analytic representation of an arc can be non-parametric, or parametric. We use the term "parametric" in the sense that the coordinate values of the points in the curve are functions of a parameter t that takes on values in the interval $(0,1)$. We use the term "non-parametric" to include all other representations of the curve. For example, an analytical-arc may be specified: indirectly as the intersection of two surfaces, or directly by giving the equation of the curve. Analytical-arcs can be lines, conics, or splines. An analytic-arc may be specified implicitly or explicitly. For example, a line in its standard form is specified by its starting and ending points and a line in its implicit form is specified as the intersection of two planar surfaces. Similarly, the general representation for a conic is as a polynomial. Often algorithms for curve fitting use alternative representations. For example, a circle is specified by its center and radius, and an ellipse may be specified by the lengths of its major and minor axes, the orientation of the major axis and its center point. A parametric-spline-segment is the parametric polynomial representation of an arc segment. A spline is represented as a sequence of parametric-spline-segments. The segments are delimited by the breakpoints along the arc. From the general object "spline" one may derive specialized instances of splines depending on the spline type, spline dimension, terminating conditions etc.

A composite-curve is an ordered list of the basic arc types, namely: point, line, circle, conic, or spline. If the composite-curve had N arcs then the added constraint is that the ith arc's endpoint is the same as the $i+1$th arc's startpoint. A digital-arc-chain is a composite-curve. A spline is also a composite-curve. The main difference between the Composite-Curve and the Linked-Curve-Segment described in the Geometric-features section is that a Composite-Curve is connected, whereas the Linked-Curve-Segment need not necessarily be connected. In other words, a Linked-Curve-Segment may just be a linked list of curve segments.

12 Surfaces

12.1 Scope

A **Surface** is a general class for describing piecewise two dimensional manifolds and different discrete samplings of them. Surfaces are used in computer vision for describing models, for the characterization of image intensity areas, for fitting parameterized functions to data, and for reasoning about visibility and geometric relationships. Surfaces can be bounded and unbounded. The curve class is used to describe the boundaries of bounded surfaces.

12.2 Conceptual Description

A partial hierarchy for surface geometries is shown in figure 11.

Surfaces are broken into a few general classes. A simple (or elementary) surface can be described by an analytic or parameterized expression. Examples are planes, spheres, and quadrics. Bounded instances of the simple surfaces are specified by range constraints on their parameters and topological representations such as the *face*(see the section on solids). Bounded versions of parametric surface patches can be specified by regions in the UV-parameter plane. The UV-parameter plane can be treated as an image for associating surface registered properties. Ribbons are treated as a kind of composite curve based upon a relation between an axis curve and a sweeping curve.

Composite surfaces are formed by connecting simple surfaces together. The general composite surface uses a relational network which describes the underlying topological relations between vertices, edges, chains, and faces. Composite surfaces can be made out of superquadrics, polygonal patches, and the different types of parametric patches. Currently, composite surfaces are only made from patches of same type. The different types of composite surfaces specialize the general relational network to describe different blending relations between patches.

Discrete surfaces are described by binary 3D-arrays with values indicating which positions are occupied in the minimum bounding rectangular prism containing the surface. This array also has a specified position with respect to a coordinate system. Discrete surfaces can be used for rapid determination of intersections, and searching operations.

The methods associated with surfaces are similar to those found with other objects such as curves and volumes: methods for geometric operations, for combining surfaces, and for mappings between surfaces and other IUE spatial objects. There are creation methods for instantiating surface objects. Among these are fitting methods for instantiating a surface to a discrete point set, such as fitting a plane, a sphere, or a polygonal mesh to a discrete 3D point set. There are also sampling methods for creating a discrete point set of positions on a surface such as regular positions on a plane or a sphere.

In some instances, the same surface can be described in multiple ways: a cylinder is can be defined as a ruled parametric patch or by a simple parametric expression. If a user desires, he can create a new class which inherits from both descriptions. There are significant issues if the attributes of one type of description are changed.

12.3 Extensions

There is a strong similarity between the required IUE functionality for curves, surfaces, and volumes and what is supported in different graphics and CAD packages (Renderman, PHIGS, the ESPRIT specification of a Neutral File for CAD geometry, and others). There are also strong differences: in computer vision the emphasis is not realistic scene generation but on such things as generating a predicted segmentation and access to objects and their relative depth ordering to reason about visibility and lighting effects. We are interested in a deep linkage with existing and developing standard graphics packages so they can be used for geometric reasoning and to generate IUE objects in addition to realistically rendering an image of a scene. For example, when an surface is displayed, we'd like to also get image registered curve and junction objects generated as a predicted segmentation. This capability would be very useful for producing data for testing and image-driven interactive creation of models.

13 Solids

13.1 Scope

Many Image Understanding techniques depend on some description of volumes of material in space and at the most abstract a 3D model can simply be space occupancy. At that level, the type of operation which is appropriate is whether a point in space is filled or empty and if filled, what kind of material is at that point. One can even define display operations at this level without any commitment to a particular specific data structure. It is also possible at this level to describe simple geometric and topological properties:

- Connected Components
- Symmetries
- Moments or other distributions of matter
- Intersection of Volumes(CSG Ops)

At a similar level of abstraction we can define the points in space which lie on the interface between different materials and temporal behaviors. Again we don't commit at this level to a specific boundary representation, but recognize that the concept of object surface is central to vision representations. At this abstract level it is possible to define many generic properties and operations. For example:

- Surface Geometry (normal, tangent plane, curvature, ...)
- Surface Properties (reflectance, roughness, ...)
- Surface Intersections

At this level of abstraction we can introduce the common notions of topology without consideration of a specific geometric representation. For example, the 2-cycle, which a closed network of surface regions, does not make

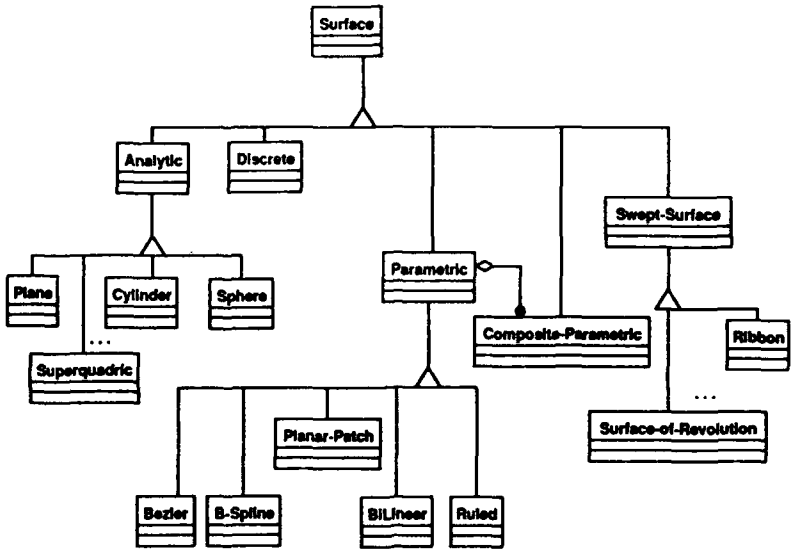


Figure 11: A partial hierarchy for surfaces.

any commitment to a specific representation but merely describes the closure of space.

13.2 Conceptual Description

These general principles are followed in the following class hierarchy so that may different types of 3D model representations can be accommodated and still permit a relatively small set of generic methods to encompass most of the operations required for vision research.

The class diagram shown in figure 12 shows the most abstract classes for a number of 3D objects. In three dimensional space it is possible to have objects which are described by zero, one, two, or three parameters, i.e. points, curves, surfaces and volumes. Parametric volume representations often arise in medical image analysis associated with tomographic image modalities such as x-ray tomography and magnetic resonance. For example one could define a tissue volume with a distribution of x-ray density described by spherical harmonics. However, we will not detail these types of three parameter volume descriptions in the current hierarchy since most current vision research focuses on surface representations or on volumes which enclose uniform solid material. Also coordinate frames and other auxiliary structures for establishing spatial relations in 3D space are covered in other sections of this document and will not be discussed further here. Figure 12 we defines two basic object forms, the volume and boundary representation. In the figure, the boundary representation is shown in more detail. The standard topological representation is defined by a series of classes as follows:

- Vertex - A zero dimensional topological element which defines a point in space where two or more entities are incident.
- Edge - A one dimensional topological element which defines a bounded curve. The curve is bounded by a vertex at each end. A closed curve is assumed to have an embedded vertex to define the boundary. Note that any type of curve can be used to represent the geometry of an edge.
- 1-Cycle - A closed sequence of edges with a common vertex bounding adjacent edges. A related notion is often introduced called the 1-Chain which is a sequence of edges, not necessarily closed.
- Face - A two dimensional surface which is bounded by one or more 1-Cycles. A multiply connected face requires an outer bounding 1-cycle and a number of interior 1-cycles. The surface enclosed by the face can be also any surface representation. If the surface is intrinsicly closed, such as a sphere, it is assumed that a vertex boundary is embedded in the surface in analogy to the closed edge loop.
- 2-Cycle - A closed sequence of faces. Here faces are joined by a common edge. Each edge is adjacent to exactly two faces.
- Block - The block is an enclosed volume of space. A block can be multiply connected and thus the block boundary is a set of 2-Cycles in a similar way to the multiply connected face.
- Object - Finally, an object is a collection of blocks which are joined at faces.

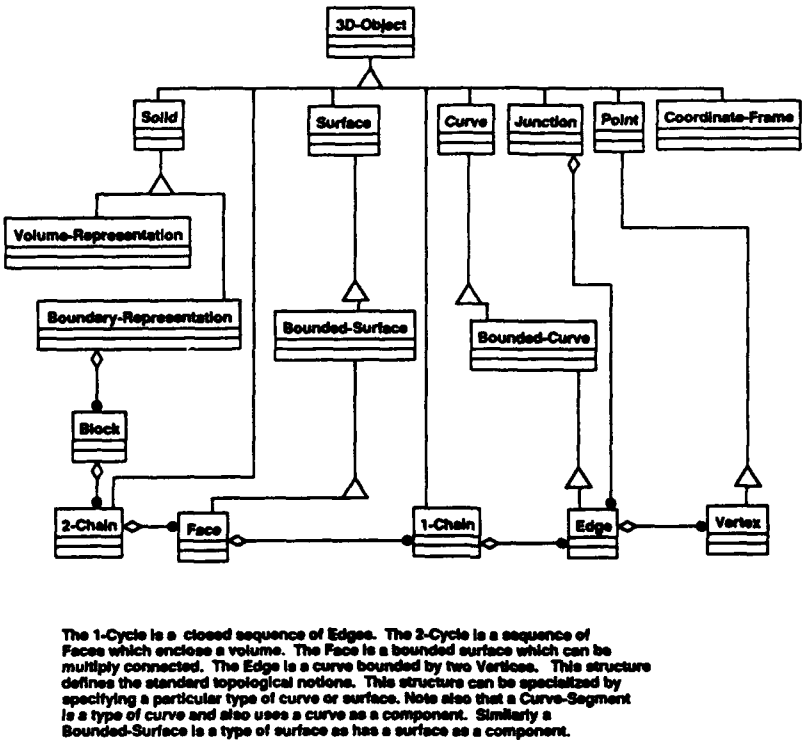


Figure 12: The top level hierarchy for 3D solids

The representation is topologically complete and can serve as a medium of exchange among various boundary representations. A major problem in exchanging boundary descriptions is the unambiguous recovery of the correct topology for the surface. There are many choices for a complete description, such as the winged-edge format, but the proposed representation is convenient for carrying out boolean operations and for constructing a boundary description from a wire frame representation. The chain data structures correspond directly to classically defined topology and chain algebra is a well developed approach to topological operations. This topological description is described in more detail in the paper by Wesley and Markowski [3]

Constructive Solid Geometry The other major solid representation is in terms of solid primitives which can be considered blocks from the boundary point of view. These primitives are combined by attachment and boolean intersection operations to form composite objects. The resulting composite object is described by a tree where the nodes of the tree represent various primitives and partial constructions and the arcs of the tree represent boolean or attachment operations. This description is referred to as the Constructive Solid Geometry, or CSG, representation.

Two examples of volume primitives used in current vision research are the superquadric and the generalized cylinder. The superquadric is a generalization of the quadric surface by providing variable exponent values on the quadric terms. The generalized cylinder is a major representational approach where the primitives are defined by a axis which can be a general space curve and a sweeping rule which defines the variation of the object cross section along the axis. For example, a cone is a generalized cylinder with a circular cross section and a linear sweeping rule along a straight line axis.

A specific object is generated by combining these primitives in a CSG tree. This representation requires that boolean intersection operations are defined for each primitive which allows quite complex objects to be defined by a short description. For example a cylinder with a hole can be defined as the subtraction of one cylinder from another. Usually each primitive is associated with a bounding box (rectangular prism) which facilitates efficient checking for the possibility of intersection.

13.3 Extensions

A standard exists called the Initial Graphics Exchange Specification or IGES for exchanging geometric descriptions. The standard is supported by the U.S. department of commerce and the initial purpose of IGES was to exchange drafting information between various CAD vendor systems. IGES does provide standards for the representation of some solid primitives and a partial treatment of a solid boundary representation. The CSG representation is more complete than the boundary representation. Post-fix and infix notations are provided for

[3] Markowsky, G., Wesley, M.A. "Fleshing Out Wire Frames," *IBM Journal of Research and Development* 24, (5), 1980.

the CSG tree operations: union, intersection and difference. The solid primitives supported by IGES are as follows:

- Rectangular Parallelpiped
- Right Angular Wedge
- Right Circular Cylinder
- Right Circular Cone [May be truncated]
- Sphere
- Torus
- Solid of Revolution
- Extruded Solid
- Ellipsoid

THe IUE committee is currently debating the relationship between IGES and the 3D solid standards needed by the IUE.

14 Coordinate Frames

14.1 Scope

The geometric relationship between sensors and scenes, among physical objects, and between pixels and the world has been a core component of the science of image understanding since its inception. Nearly every IU system makes use of coordinate systems and transforms either implicitly or explicitly. The multitude of representations that have been devised, some of which incorporate arbitrary conventions, has been a key obstacle precluding the transfer and sharing of code and results. In this section we describe the representations of coordinate systems and transforms that are to be provided by the IUE. It is expected that these constructs will be employed in many areas of the IUE, such as image features, 3D models, and sensor objects.

14.2 Conceptual Description

We begin with definitions of our terminology:

Coordinate System: A coordinate space, in the mathematical sense. It is represented in the IU environment by an instance of a coordinate system class.

Coordinate: The coordinate(s) of a point are represented by a series of numbers, and are implicitly associated with a coordinate system.

Coordinate Transform: A specification of a mapping between two coordinate spaces. It is represented in the IU environment by an instance of a coordinate transform class.

Conceptually, the relation between coordinate systems and coordinate transforms can be expressed by a directed graph in which a coordinate system is represented by a node and a coordinate transform is represented by a directed arc between two nodes. Both coordinate systems and coordinate transforms are represented by instances of object classes. The classes of transforms that can relate two coordinate systems is governed by the classes of those coordinate systems.

Figure 13 shows the class hierarchy for coordinate systems. The leaves of this tree indicate the classes that can be instantiated by an IUE user to specify the coordinate space that he chooses to use. The IUE programmer can extend the hierarchy by adding new class definitions to allow use of other coordinate systems. As can be seen in the figure, both Cartesian and non-Cartesian coordinate systems are supported, including a number of coordinate systems commonly used for geographic purposes (such as UTM and Geodetic).

These coordinate systems will contain slots appropriate to their class, the most important ones being:

- Dimension — The dimensionality of the coordinate space.
- Related-coordinate-systems — A data structure that specifies the coordinate transforms that have been defined for mapping coordinates in this system to other coordinate systems. Collectively, these constitute the coordinate transform graph that relates all coordinate systems.

A coordinate system object supports a number of commonly useful methods, including:

- Find-transform-to — This method returns a transform that can be used to map coordinates from this coordinate system to a second coordinate system, which is passed as an argument.
- Find-transform-from — This method returns a transform that can be used to map coordinates from the coordinate system passed as an argument.
- Translate, rotate, scale, ... — A collection of methods that allow the relation between coordinate systems to be modified, using operations appropriate to the class of the coordinate system.

Figure 14 depicts the class hierarchy for the coordinate transforms that can be used to relate coordinate systems within the IUE. This hierarchy includes common matrix transforms, a collection of transforms for mapping to geographic coordinate systems, and other specialized transforms such as those represented internally by quaternions and by analytical functions.

Slots that are defined for all classes of coordinate transforms include

- From-coordinate-system — The domain for the mapping specified by this transform.
- To-coordinate-system — The range for the mapping specified by this transform.
- Inverse transform — A transform expressing the inverse mapping (optional).

All coordinate transform classes support at least the following methods.

- Transform-point — The method that actually computes the coordinates in the to-coordinate-system, given a set of coordinates in the from-coordinate-system.
- Compose-transforms — A method which collapses a sequence of several transforms into a single one.

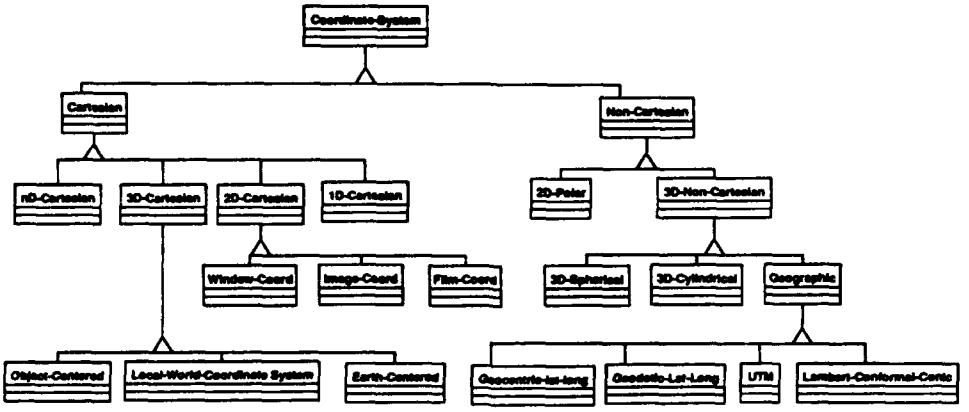


Figure 13: Class Hierarchy for Coordinate Systems

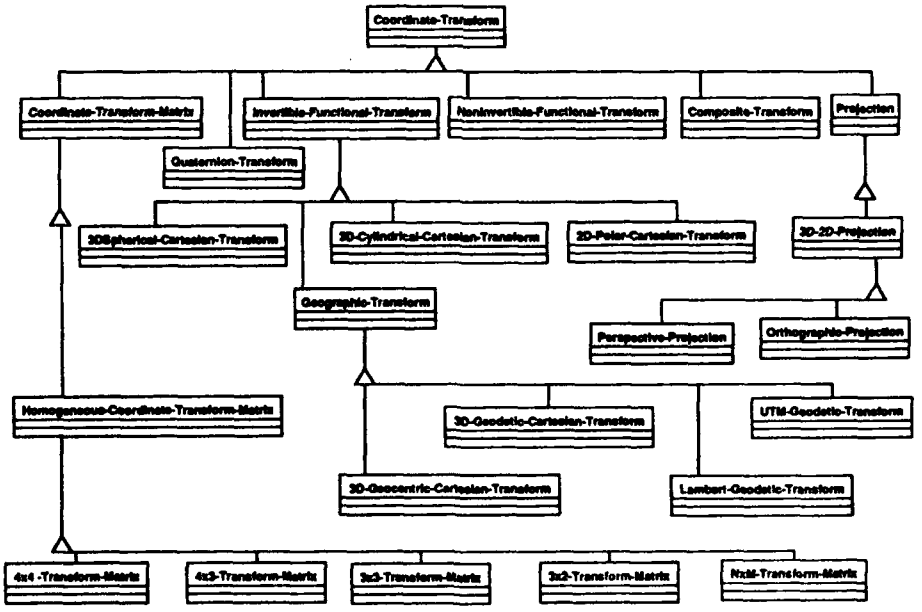


Figure 14: Class Hierarchy for Coordinate Transforms

14.3 Applications

It is anticipated that coordinate systems and transforms will be used to relate all geometric objects within the IUE. An example of how these might be used is depicted in Figure 15. Coordinates in any two coordinate systems can be related by composing the transforms found by traversing the graph.

For example, the UTM coordinates (CS-5) of a point can be converted to latitude-longitude (CS-4) using transform CT-4. These coordinates can, in turn, be related to a local Cartesian coordinate system (CS-3) and then to either film (CS-2) or image (CS-1) coordinates. Other coordinate systems can be related to this network using additional transforms as exemplified in Figure 15. The IUE will provide the mechanisms to freely move between coordinate systems by traversing this network.

14.4 Extensions

Speed of execution is essential if coordinate transformations are to be useful within a computer vision system. While our main emphasis has been to provide for a general and extensible collection of coordinate systems and transforms, it is necessary to insure that the requisite computations can be carried out swiftly. Several mechanisms to facilitate this will be incorporated as the IUE design is fleshed out:

- Composition and caching of long chains of coordinate transforms into a single transform. Additional machinery must be provided to keep the coordinate transform graph consistent when multiple paths exist.
- Access to hardware accelerators. Additional coordinate transform classes can be defined that allow access to computations performed in hardware. The idiosyncrasies of their use should be largely hidden by the specification of appropriate coordinate transform classes.

15 Constraints

15.1 Scope

The use of constraints in Image Understanding research is quite diverse and in many ways the proper treatment of general constraints impinges on the general AI problem. The following summarizes some the ways in which constraints can be defined and used.

Label Assignment The label assignment problem can be viewed as a constraint satisfaction problem and be solved by heuristic search methods. The idea is to consider a partial label assignment as the node of a search graph and the cost assigned to a node is related to the degree with which it satisfies the constraints between currently assigned labels. Recent work in generic object recognition (Levine et al) using GEONS applies this type of constraint processing.

Constraint Propagation Networks The representation of constraints as a network has appeared most often in neural network research. A typical example is the Hopfield net where nodes have a particular input/output constraint. In the case of the Hopfield net, the output of a node is activated if the balance of the weighted inputs is positive. Such networks are often solved by relaxation methods where node states are adjusted until a consistent overall network state is achieved. Another approach to such networks is the use of integer linear programming methods.

Geometric Constraints Perhaps the most central use of constraints in IU research to date is the application of geometric constraints. A significant example is the ACRONYM system which represents object models in terms of generalized cylinder components. The configuration and shape of the components is restricted by a set of geometric constraints which are defined for a particular generic object class such as an aircraft. Constraints can also be applied to camera viewpoint. In ACRONYM these constraints are defined as symbolic inequalities and solved using a symbolic rewriting algorithm called SUP-INF.

The advantage of such use of constraints is that known relations which must exist between entities can be realized, even in the presence of empirical measurement error or numerical imprecision in forming the model instance. For example, a measured angle between lines might be 89.6^{o} but it is risky to conclude that the lines are intended to be perpendicular. The symbolic constraint relations are always directly specified in the model representation and do not have to be inferred from the imprecise geometry of a model instance. A final advantage is that a wide range of geometric shapes can be expressed by a relatively compact constraint description.

15.2 Conceptual Design

The following object designs are primarily focussed on geometric constraints but the approach appears to be generalizable to handle the other examples just discussed. There are three basic object classes: symbolic-entities, constraint-relations and the constraint system. These object are related as illustrated in figure 16.

The **symbolic-entity** is a symbolic form of an IU class such as a plane, transform, or curve. The key parameters or attributes of the class are represented as symbols rather than fixed numbers. For example the orientation of the surface normal of a plane can be represented as three variable symbols. There also may be equations which are part of the definition of a particular IU class. For example, an ellipse is defined by $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$. The symbolic enitity also contains pointers to the constraint relations in which it takes part. In general the symbolic-entity hierarchy will mirror the non-symbolic class hierarchy. This duplication seems desirable so that more conventional applications do not have to carry around the baggage of constraint representations.

The **constraint-relation** is also a symbolic structure which relates entities in a network structure. It is not unreasonable to consider a symbolic-entity as a unary constraint-relation, but I have chosen to keep them distinct for clarity. The constraint relation provides pointers to the related enitities as well as the equations involved in defining the constraint. For example, if two

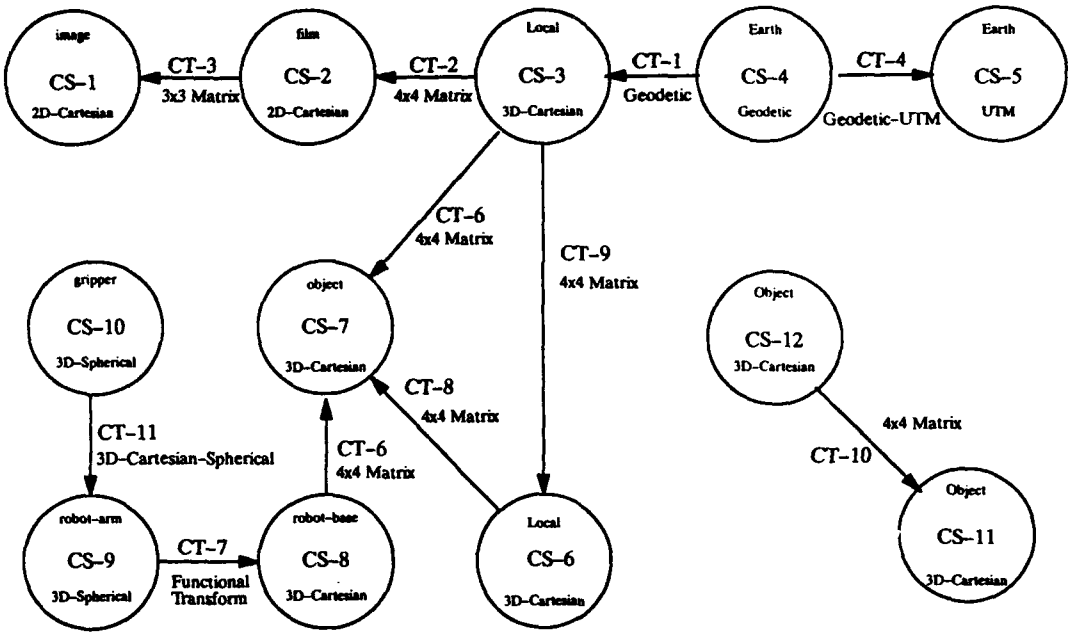


Figure 15: Example of possible relationships among coordinate system instances and coordinate transform instances

vectors V_1 and V_2 are constrained to be perpendicular then the constraint equation $V_1 \bullet V_2 = 0$ is constructed by the corresponding constraint-relation class instance.

The **constraint-system** consists of a constraint relational network and the associated entities. The main function of the system is too collect together all of the constraint equations and symbolic variables and compile them into a form for efficient solution. In the case of geometric constraints, all of the equations are polynomials, and therefore the solution of the constraint system corresponds to finding the roots of a multivariate polynomial equation set. It is also often the case that the constraints must be satisfied in the context of data measurements such as image boundaries or region moments, as in the case of "snakes". In these applications the solution of the constraint system corresponds to constrained minimization, or non-linear programming. The standard approach to this in the IU literature is the Levenberg-Marquat algorithm [4]. In any case, the constraint system is responsible for collecting the appropriate data and forming a cost function for the minimization.

Another function of the constraint-system class is to collect all of the information about the constraints and symbolic-entities needed to form an efficient representation for solving the contraints. This process of collection is reasonably called parsing the constraint network. In the case of geometric constraints, the equations and variables are collected to form a jacobian matrix which is the primary representation used in the Levenberg-Marquat scheme. The equations can be differentiated numerically or symbolically, as appropriate.

[4] "Numerical Recipes in C," W. Press et al Eds, Cambridge University Press, 1988

16 Sensors

16.1 Scope

These objects provide a number of important IU capabilities including generation of synthetic data via rendering, reasoning about sensory events, image/data filtering. Related to sensors are the classes Energy-Objects, and Filters described in the following sections.

The following summarizes our view of the "sensing" process: *Physical sensors take energy from the world, apply some "focusing" and filtering mechanism on their energy input, convert one form of energy into another, possibly filters this new energy form, and finally a conversion into a final discrete form. The energy input to the sensor is a (generally) complex interaction of various energy-sources (some possibly controlled by the sensor) and the objects in the world/scene space. A sensor input might also be the output of another sensor, (e.g. a flat-bed scanning of a photograph of a scene).*

Sensors are thus a mapping from one space (energy inputs) to another space (discrete energy outputs). Sensors can thus be cascaded to form other sensors. In particular we believe we should support the view of logical sensors wherein a sensor's input can be the output of one or more other sensors. Thus a stereo sensor could be defined which takes as input the output of two edge-detecting sensors which each took as input the output of a camera sensor. The output of sensors is not restricted to image like objects, but rather may be any IU object.

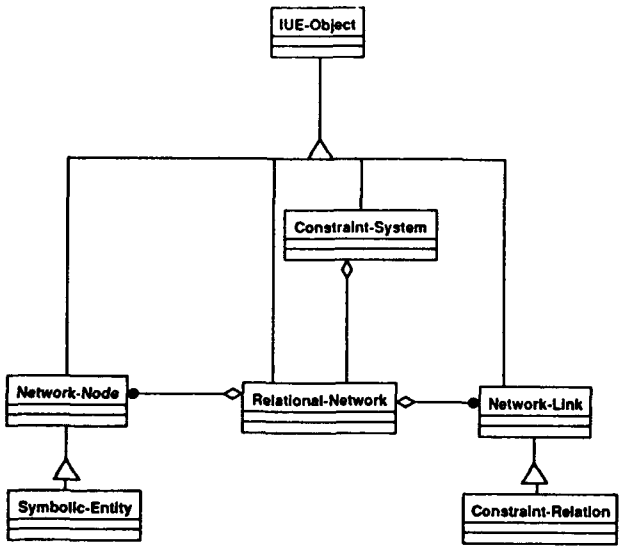


Figure 16: The top level constraint object hierarchy.

Sensors will interact with just about every IU object. The sensing of synthetic scenes will involve all just about all possible object models which can exist in the world and produces images or IU objects. The sensors naturally involve coordinate transforms, and calibration procedures will likely require the use of the constraint models.

The issues we must address with sensor and energy objects are two fold:

1. what operations needed to be supported on such objects to facilitate IU research, and
2. at what level of detail should the process(es) be modeled.

Note that the answer to the second issue affects the first in that the same operation may need to be provided with respect to various "models" which will generally affect it implementation (and computational cost). These two questions will be addressed in the following discussion.

16.1.1 Operations of sensors

We begin with the general operations, and then discuss what will be included in the initial system. Sensors are composed of objects which provide the functionality of the basic operations of the sensor. These components are shown in figure 17.

As pointed out before, one of the operations that the sensor section is to provide is rendering. It is emphasized that rendering for image understanding purposes has quite a different emphasis than graphics rendering. In the case of graphics, the purpose is to generate a realistic looking scene. In image understanding research the goal is to accurately model the image formation process to that various aspects of image analysis algorithms can be evaluated.

To be more specific, we view rendering as a special case of the forward use of a sensor, i.e. mapping from a world/scene into a data object. Because this forward mapping view is so prevalent we have defined a logical subclass called **Transducers** which are exactly those object supporting only the forward mapping operations of sensors.

16.1.2 Transducers

The fundamental operation of transducers is (logically) to ask how changes in scene/world "map"(project) into the image/object space. For notational convenience this single fundamental operation is decomposed into two operations, *Transduce* which is supposed to compute a "likely" image/object given the world/scene. A slightly more aggressive method, *Data-certainty* will allow one to ask about the certainty of (subparts of) the image/object given the current (hypothesis of) the scene/world space.

Transducer: scene/world → image/object.

The remaining operations will be those needed to support these two basic operations. These including setting parameters which affect rendering, setting parameters which affect certainty calculations, including a general calibration which may simultaneously affect both the world/scene and rendering/certainty parameters.

The objects will come with defaults which build up the rendering and certainty calculations by a concatenation of methods applied to the components of the sensor. For example the default rendering might be constructed as a ray-tracing of the scene which provides a super-sampled input to the energy-transfer-system (which actually uses the reconstruction-prefilter to convert to whatever sampling rate it desires). The energy-transfer-system would outputs a function which is then filtered by the spectral-response, and then sampled by the output-sampler.

Since each of these might involve "arbitrarily" complex filtering the full functionally of a sensor should be achievable in any single step (e.g. the output-sampler could do everything). However, this decomposition of the model was constructed as a sequence of filters to reduce implementational burden on the user and to allow conceptual (and implementational) modeling of sensors by a collection of simpler standard components. In cases where each filtering step is a linear filter (and the noise models are simple enough), the system will compose this sequence of filtering operations into a single linear filter significantly decreasing the computational complexity. This will allow reasonably efficient implementation in most cases.

Automatic definition of the data-certainty method is considerably more difficult and will assume that each filtering component is from the subclass of noisy filters which provide an underlying certainty function for their mapping, and that the noise is *pointwise independent*. Thus the certainties can be directly composed. For the general case where inter-dependencies are assumed, the user will have to supply the certainty-function directly.

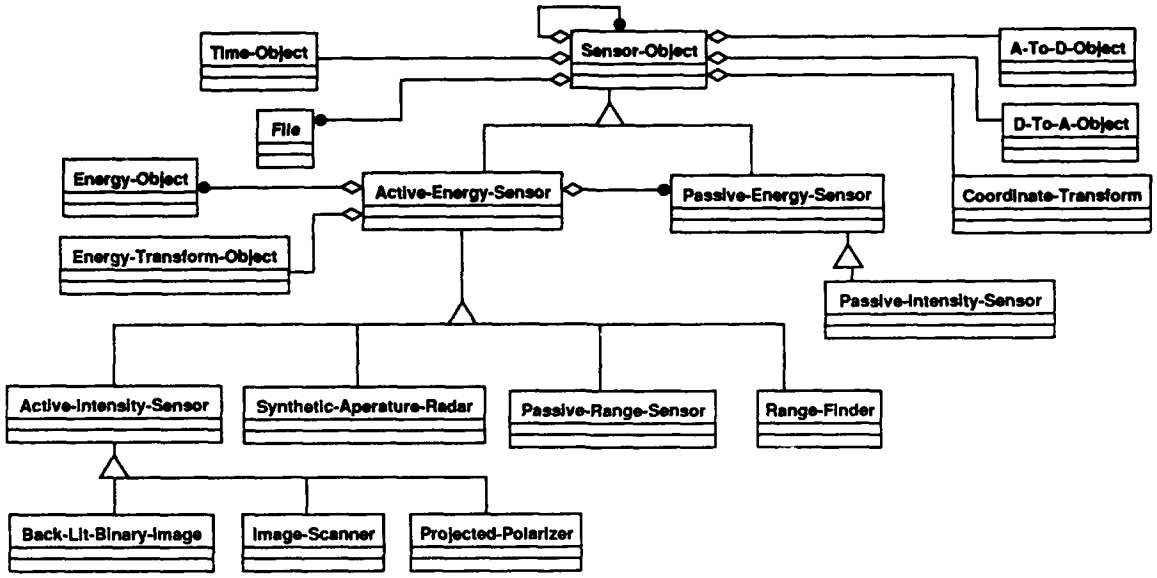


Figure 17: The top level sensor hierarchy.

16.1.3 Image Sensors

Sensors are a generalization of transducers where we consider the true bi-directional mapping nature of physical sensors. They provide the basic operations *Sense* which returns a **sensor-data-object** based on a "likely" image/object (as opposed to *Transduce* which directly returns an image/object). The **Sensor-Data-Object** provides a mechanism for mapping from the image/object back into the world/scene. For this inverse mapping there are two basic operations, inverse-point-projection, which returns a volume in the scene which would map into a given image/object point. This inverse mapping is based entirely on the geometric aspects of the sensors subparts. A more costly operation will be *Scene-Certainty* which will allow one to ask about the certainty of a scene/world object given the sensory data. Note that the problem of computing *Scene-Certainty* method is generally ill posed and will involve a number of assumptions. Related to the *Scene-Certainty* method is the method *Invert-sensing* which will attempt to return a likely image/object which, when passed through the sensor, would account for the data. Obviously this last method is not even a well defined mapping (let alone mathematically well-posed) since we have no restrictions on the space on input world/scenes and can not hope to have a unique definition. In general this method will need to be provided by the user, though a default will be provided which assumes the world is a frontal projection of a plane with an "image" painted on it, in which case this method is simply image deblurring and resampling.

We want to emphasize that all of the certainty calculations associated with sensors *need not by physically accurate*. Rather, they are mathematical models of the certainty and hence their relation to reality depends on the faithfulness of the model. The goal is to provide a way for IU researchers to test their algorithm using synthetic data generated according to a "pure" model (e.g. lambertian surface with point light source viewed orthographically with a blur-free camera and white-noise of a given level) or increase the realism and also test under different more realistic conditions (e.g. Torrance-sparrow type partially specular objects with multiple extended light sources, viewed through a thin-lens with depth of field, focus and chromatic aberration affects with position dependent exponentially distributed noise and a simple model of CCD charge bleeding.) In addition we hope to provide a framework (via the bidirectional certainty mappings) which will be useful to IU researchers engaged in sensor modeling and using sensor models in IU applications. By providing a consistent interface with respect to certainties higher level models that want to incorporate this information into sensor fusion tasks can work with a very wide variety of sensors without changing their interface to the sensors.

16.2 Level of modeling

We believe that the mechanisms proposed are sufficient extensible to allow many levels of detail to be effectively pursued. The question as to what levels will be provided in the basic IUE is a function of manpower devoted to the sensor section for initial development, the willingness of researchers to allow incorporation of existing software for certain modeling problems, and the effort necessary to convert or interface to existing software. We remind the reader that the system is not intended to be real-time and some of the proposed operations, especially if modeled at a very low level, are expected to take considerable compute power/time.

It seems clear that we should provide tools to allow

middle level researchers to build upon lower level models, when the wish to, but also provide an efficient implementation of an abstract model closer to their level of use. E.g. we have proposed a class of **Grayscale Camera** which returns images, but does not get into such detail as photo-site layout, charge bleeding, photo-site spectral characteristics, etc. Thus a **Grayscale camera** equipped with a pinhole simple lens also Transduction in a fraction of a second, while a CCD camera with a zoom-lens may take many hours. From the users point of view, all methods applicable to a Grayscale camera are also supported by a CCD camera, only slower.

16.3 Energy Objects

These objects provide the functionality for the IUE to represent and reason about energy-waves (light). Most of the impact of this class has to do with rendering (synthetic sensing), though the objects could also be used directly by other methods.

There are two basic subclasses of energy-objects: wave-bundles and energy-sources. The first category provide a low level way of representing a generalization of light-rays and come in many flavors, including monochrome rays, RGB rays, multi-spectral rays, polarized multi-spectral rays, light pencils, and light volumes. These will allow IUE users to do their own ray-tracing of scenes (if needed for advanced applications though this will be rather slowly compared to traditional ray-tracers). More commonly they will be used for reasoning about small portions of an scene in a more sophisticated manner (e.g. phase computations for coherent sources in regions the neighborhood of depth edges). In the initial IUE implementation, there will be few of the potential flavors actually used, though the framework and methods will be defined for most.

The second class, energy-sources, is a generalization for all other characteristics of the way object absorb/emit/reflect energy. All objects, which are intended to be rendered, will subclass off energy-sources. A regular (reflective) object will simply have an energy-distribution that is functionally related to its energy input and has total power output $\leq$ the total power input. The basic operations allow one to affect parameters, including energy (spectral) distribution, total power, BDRF. In addition there are methods to allow incremental evaluation and update of a scene containing energy-sources, and for computing the energy flux at a point. The proposed sources include point sources, planer extended sources, sun-light, laser-sources, patterned sources and environment-mapped sources. We further propose a energy-volume source which may be useful for modeling fog or smoke affects.

While it may seem counter intuitive to have all objects be energy-sources, it is actually more realistic. When interreflections are taken into account, each surface acts, in one sense, like an energy source. By making a single object from which to derive all reflective properties, we can insure that for the same scene and sensors, those researchers wishing to include interreflections into their rendering can be accommodating. To make the rendering process efficient (for simple rendering models), the scene objects will actually have a list of "active" sources. These are sources which are active in the sense of outputting more energy than they receive.

16.4 Filters

Both the sensor objects and the energy objects make heavy use of filter objects. This class provides a means for "filtering" functions (discrete or continuous) in a consistent manner. The concept of filtering is too extensive to come up with a efficient method for implementing any "filtering" mechanism in our OOPL. This class takes filtering sub-topics which are important for IUE and attempts to handle them.

A very important (from an implementational efficiency point of view) category of filters are Linear filters. The base-filter class provides a slot to determine note if a filter is linear. A the "linear" property of a filter is never checked, and every filter may operate as though it were linear, it is up to the IUE user to verify linearity and to check the isLinear slot. The base-filter class also provides for such operations as building a filter-sequence (a sequence of filters which are called in order), and manipulating the linear filter characteristics of the filter such as specifying the impulse response, computing the impulse-response directly, setting the sampling rate for the FFT representation, computing the MTF from the impulse-response, specifying the MTF directly, and computing the impulse response from the MTF. The base-class is virtual, no instances are allowed. One can specify, for linear filters, if the filter should be applied directly, or if it should be used in the frequency domain (in which case the the MTF is really the definition of the filter). For higher dimensional filters, there is the ability to specify a separable form for linear filters (but there is no way to compute them automatically). Finally, there is a compose-linear-filter operation which will allow one to build a new linear filter directly from a sequence of others (by up-sampling their MTF's to consistent rate and doing pointwise multiplication).

The four basic subclasses here are the DtoA filter, the AtoD filter, the DtoD filter and the AtoA filter. These provide a consistent interface to allow transformation between digital and analog forms for data and to allow building of complex transformation from simpler ones. To allow flexibility, the Filters are applied to either fixed dimension functions with a consistent calling mechanism or to arrays. Their output can be either form except, obviously, that analog outputs cannot be arrays. Note that since images are subclassed from arrays filtering images is straight forward. While we believe the AtoA filter class with be rarely used (filtering in functional for is generally hard without knowledge of the internal representation (which is not specified), and a cascade of AtoD and DtoA is better specified in that form.

17 Other Extensions

In spite of the rather extensive object specifications in the preceding sections, there are a number of areas which require further investigation. These areas will be further defined in the final IUE specification.

17.1 The Spatial Object

In developing the various class structures we have noticed that common spatial attributes and operations arise over and over again. Examples are,

- spatial dimension
- coordinate frames
- coordinate transformations
- intersection, e.g. point-on-line
- containment, e.g. point-in-polygon

We intend to capitalize on this abstraction to streamline the classes associated with spatial description such as curves, surfaces and solids

17.2 Spatial Indices

Many IU algorithms gain efficiency through the use of spatial indexing methods. We have already mentioned the Hough array as a classic example, but there are many others which should be available in the IUE. The following is a partial list to illustrate the concept.

- k-d Tree
- Minimal Spanning Tree
- Quadtree
- Octree
- Delunay Triangulation
- Various Hashing Schemes

17.3 Statistical Operations

We recognize that statistics is a major influence in many IU approaches and algorithms. We intend to provide a basic set of objects and tools for using statistical methods. It is likely that a Kalman filter infrastructure will be specified since this approach has experienced considerable popularity in recent years, particularly in active vision work.

17.4 Object Oriented Databases

Not reflected in this paper are some preliminary studies on the emerging technology of object oriented databases. There are already commercial packages available for C++ and we are studying various approaches to make such facilities available for Lisp. The availability of a persistent database for objects will considerably improve the efficiency of rapid prototyping, since currently a great deal of effort is expended on file I/O and specialized databases for efficient retrieval of IU structures.

17.5 Data Standards

We have already discussed the problem of import and exchange of IU data and results between other systems and the IUE. Perhaps the most important role of the IUE committee is to define and maintain a data exchange standard for IU data structures.

A preliminary design for the exchange mechanism has been defined based on a list structure. The most important issue is the definition of keywords for common IU attributes and data structures. It is planned to defined file structures which carry the data definition along with the data so that exchange interfaces can be easily implemented. One, perhaps over-optimistic, idea is that the IUE specification LaTex macros can be used to generate the data exchange formats directly.

References

[1] A. Hanson and L. Quam. Overview of the SRI Cartographic Modeling Environment. In *Proceedings DARPA Image Understanding Workshop*, pages 576–582, Cambridge, MA., April 1988.

[2] C.C. McConnell and D.T. Lawton. IU Software Environments. In *Proceedings DARPA Image Understanding Workshop*, pages 666–677, Cambridge, MA., April 1988.

[3] Thomas D. Williams. Image understanding tools. In *Proceedings 10th International Conference on Pattern Recognition*, pages 606–610, Atlantic City, NJ, 1990.

The RADIUS Common Development Environment

J. L. Mundy and **R. Welty**
General Electric CR&D
Schenectady, New York

L. Quam and **T. Strat**
SRI International
Menlo Park, California

W. Bremner, M. Horwedel, D. Hackett and **A. Hoogs**
General Electric M&DSO
King of Prussia, Pennsylvania

ABSTRACT

RCDE is a software environment for the development of image understanding algorithms. The application focus of RDCE is on image exploitation where the exploitation tasks are supported by 2D and 3D models of the geographic site being analyzed. An initial prototype for RCDE is SRI's Cartographic Modeling Environment. This paper reviews the CME design and illustrates the application of CME to site modeling scenarios.

1 Introduction

1.1 RADIUS and RCDE

Research and Development for Image Understanding Systems, or RADIUS, is a DARPA-funded project to develop and demonstrate new image exploitation capabilities, based on image understanding techniques. The central theme of RADIUS is the use of three dimensional site models which provide the contextual information needed by image understanding algorithms. The premise of RADIUS is that by registering a site model to an image, the identification of objects and the detection of significant change is much more reliable than by totally bottom-up procedures, such as image subtraction.

In order to demonstrate these model-supported image understanding techniques, a flexible software environment is being developed called the RADIUS Common Development Environment, or RCDE. RCDE provides a set of basic data structures and algorithms to enable image understanding algorithms to be demonstrated on application images so that individual researchers do not have to repeat the development of these basic tools. RCDE also provides the infrastructure to handle large images and a full set of geospecific coordinate representations. Tools are also provided to construct geometric models from image data and to exploit the constraints provided by terrain and sun ray geometry.

The goal is to place RCDE in many IU labs so that groups interested in IU research focused on image exploitation can conduct experiments in the context of a realistic application with minimal software development overhead. RCDE will provide basic data structures and processing operations that often arise in such applications as, object recognition, change detection and image perspective transformation.

Consider a typical example where an IU researcher may have an interesting new idea for image feature extraction. RCDE will provide

a substrate for developing an evaluation experiment for the new feature extraction method. The new feature extraction results can be used by other experimental systems build on RCDE and the performance of the new features can be evaluated in terms of improved performance of these other systems.

2 History

The SRI Artificial Intelligence Center Perception Group has been conducting computational vision research in cartography and photo interpretation for over 15 years. Toward this end, SRI has developed two major software tools: ImagCalc, and the Cartographic Modeling Environment (CME).

ImagCalc is a general-purpose system for image manipulation that supports a convenient user interface to a rich collection of image operators and interactive tools. Of particular importance is an underlying (internal) image representation that efficiently handles the manipulation and display of very large images (larger than 4k x 4k pixels).

CME is derived from ImagCalc and has been designed for interactively deriving 3-D models of the world using real world imagery and geometric constraints. The model derivation process can be thought of as inverse computer graphics, i.e., generating the 3-D models using images of the world, rather than generating images using models of the world. CME combines the use of monocular and stereo imagery, multiple images of greatly differing geometries, digital terrain elevation data (DTED), and illumination (sun) models in a framework that allows the user to interact with wire frame models of world objects simultaneously overlaid on all of the image sources. The illumination models allow the derivation of object heights from their shadow lengths as well as enabling objects emplaced in a scene by the user to be shaded realistically in accordance with the sun angle in the basic image being displayed. The resulting world object models and terrain models can be used to render individual new images at arbitrary camera positions, and sequences of images along arbitrary flight paths. Both the terrain and object models can be rendered using texture maps derived from actual imagery.

Both ImagCalc and CME are currently implemented only on the Symbolics 36xx series LISP Machines. ImagCalc and CME each consist of approximately 60000 lines (2 million bytes) of source code.

ImagCalc has been licensed to approximately 34 sites, including 12 university sites.

2.1 CMEE Project

Recently, GE's Military and Data Systems Operation (M&DSO) and GE's Corporate Research and Development(CRD) have teamed with SRI to further develop the capabilities of CME and port the system to a Unix platform. The project, called CMEE for CME Enhancement, is focused on the documentation of CME and on a formal specification of the requirements and design of RCDE. The CMEE project is also exploring issues that involve the use of both Lisp and C programming for the development of image understanding systems using CME.

3 Architecture

The architecture of CME is shown in figure 1 which illustrates the scope of the system. There are four main entities which can be constructed and manipulated by the system.

Images CME can represent a large variety of image data types and a novel image address mapping scheme enables efficient access to large images.

Image Operators A wide range of image processing operations are available and accessible by menu or by a high level Lisp programming structure.

Image Display CME supports fast roaming and zooming that is comparable in functionality but somewhat slower in performance than analogous film-based light table manipulations.

Windows The window system provides a large spectrum of geometric transformations and graphics functions for image and wireframe display.

Object Models A number of object families are available such as superquadrics and

house shapes. 3D curves and ribbons are also supported.

Camera Models CME provides tools for modeling perspective image projection as well as more specialized camera models such the satellite pushbroom sensor. These models are used to relate the 3D geometry of objects with the 2D geometry of image features.

Terrain Models Terrain is represented in CME as a triangulated mesh or as a simple 2D array of elevations. Terrain data is necessary to portray a realistic 3D site configuration.

Site Model The position and orientation of the CME entities can be represented in a common coordinate frame. A local coordinate system is called the site. Geocentric and geodetic world coordinate frame are available and geolocations can be consistently determined in a number of common coordinate systems.

Rendering A site model can be displayed as a realistic image view by the application of phototexture onto the 3D model faces and the terrain facets. A sequence of views is used to generate a realistic "fly-through" of the site.

User Interface All of the CME functions can be accessed through a menu-driven user interface. An interactive programming environment is also available by executing interpreted Lisp functions. A history mechanism keeps track of cached functions for efficiency and interactive programming.

4 Scenarios

The functions of CME are probably best illustrated by a number of sample scenarios. In this section we consider three examples which exercise a reasonable fraction of the system.

4.1 Image Exploitation

ImagCalc is a general-purpose image manipulation system that forms the basis of CME. This scenario demonstrates how ImagCalc aids the user in extracting useful information from grey-level aerial images. Figures 2 through 6 illustrate the results.

First, images must be loaded into ImagCalc from the file system. Selecting the **Dired** menu (DIRectory EDit) allows the user to choose a file system directory, retrieve image files, and place them in an ImagCalc pane. The upper two panes of Figure 2 contain images that were retrieved from a file; the upper left pane shows an overhead image of an airport, while the upper right pane shows an oblique view of the terminal area.

The lower two panes of Figure 2 illustrate the results of image roaming and zooming operations. The lower left image has been zoomed out, and the lower right image has been zoomed in. Two types of zooming are available: the *Fast Zoom* commands perform a pixel-replicated zoom for general viewing purposes, while the *Zoom* commands use interpolation for greater clarity. A number of other commands are available for scrolling or repositioning an image on the pane. For example, the **Recenter** command moves the selected point to the middle of the pane, while **% Reposition** is used to move the image about the pane by selecting the relative position within the entire image.

Images can be manipulated with respect to their grey-level distribution (histogram), as illustrated in Figure 3. The **Contrast Stretch** operation under the **Enhance** menu computes a histogram of an image (lower left pane) with adjustable upper and lower adjustable thresholds. By interactively adjusting the thresholds with mouse clicks, the user can stretch the histogram interactively. The lower left pane of the figure shows the operation in progress. A green line representing the lower threshold can be seen near grey level 190 of the histogram, and the upper threshold is set at 256. Grey levels above and below the thresholds are mapped to white and black, respectively, while the remaining histogram is stretched to the original range. To facilitate interactive processing, the **Contrast Stretch** command modifies the color map directly, altering all image views on the screen; this effect disappears when the operation is complete.

Figure 4 illustrates the **Window** command. The user can select a region of interest in one image

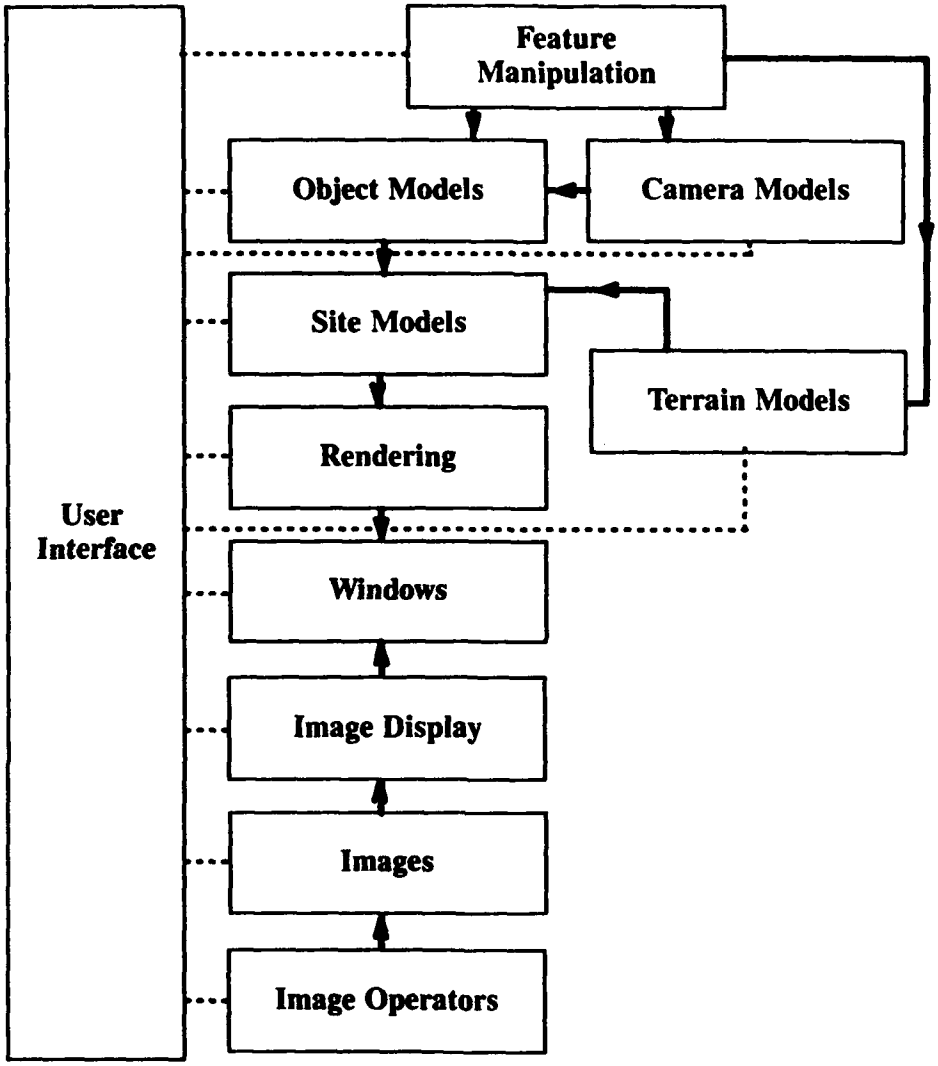


Figure 1: The architecture of CME. CME provides a closely integrated structure for the representation of images, 3D objects, terrain and cameras. These elements are the basic tools needed to support image exploitation.

and paste that region into another pane. Note the green region of interest box in the bottom of the lower right pane. The resulting pane has also been zoomed to examine its detail.

This new area of interest (the small airplane) is then used in Figure 5 to generate a surface plot. The Plot 3D command plots the image grey levels as a depth map and allows the user to interactively vary the angle of view. The result is shown in the lower right pane of the figure; the surface is viewed from the lower right corner of the airplane image. With some difficulty, you can see the green line on the image which indicates the view direction.

Figure 6 illustrates finding the outline of the small airplane using edge operators. The image is first filtered using the $\nabla^2 G$ operator, which is implemented by the Difference of Gaussians command under the **Enhance** menu. Then the Zero Cross Image command identifies all zero-value pixels, which are formed at edge crossings when using $\nabla^2 G$. The user can also modify the Zero Cross Image threshold to reduce the effects of quantization. The resulting airplane outline appears in the lower right pane of the figure.

In summary, this scenario shows how the ImagCalc portion of CME can be used to manipulate digital images. The demonstrated operations include roaming and zooming around an image, contrast stretching, cutting and pasting sections of images, generating depth maps, and detecting edges. ImagCalc forms the basis for CME's user interface and is a powerful image processing environment.

5 Scene Modeling

The CME Object system allows the user to create geometric objects, define their characteristics, and view the results. This scenario demonstrates how to create a scene of objects, and then view the scene three ways with three separate cameras. Figure 7 illustrates the scene modeling scenario.

The first step in building an Object model is to create a camera model for the scene. Starting from a blank CME pane and the default Bucky menu, select the **View Menu** command. From the subsequent menus, choose New View Transform and then New Blank View Transform to initialize the camera model. A cartesian coordinate origin appears in the center of the pane and a small box that indicates the camera's presence appears in the lower left corner of the pane. What the user sees, therefore, is a perspective camera's view of the world; the CME pane is the camera image plane.

Objects are then created to populate the scene. In this example, a house, a cylinder (silo), and a half-cylinder (quonset hut) model a farm, while a set of linked ribbon segments represents a road. The objects are instantiated using the **Create Object** command, then altered or moved using the menu commands unique to each class of objects. Modifying the object parameters is intuitive and efficient when using the Bucky menus. Alternately, each object, when selected by the mouse pointer, can be modified through the who-line Menu command, which causes a parameter window to pop up for editing the object parameters directly. For example, the roof pitch of a house defaults to 0.5; the pitch can be changed interactively using the mouse, or directly by typing a new value in the parameter window.

To show three views of a single scene, it is necessary to copy the view transform to the lower two panes. Choosing Exact Copy instead of New Blank View Transform allows the user to copy the existing scene (transform model and all the objects) to the lower two panes. All three panes then represent separate views of the same model.

Although the effect is not visible in the figure, objects are highlighted in a different color when selected via the mouse. Therefore, when an object that appears in multiple panes is selected, all views of that object are highlighted. This feature illustrates how CME is an excellent example of object-oriented software design.

The three panes now contain identical representations of the modeled scene as viewed with three distinct cameras. The default camera parameters produce the perspective view shown in the upper right pane of Figure 7. The views shown in the lower panes are obtained by moving their cameras to another viewpoint. Two coordinate systems are available for moving the camera interactively: scene-centered and camera-centered. The Azimuth/Elevation command al-

lows the user to rotate the camera about a point fixed in the scene, and is intuitive to use. Camera-centered translations and rotations (e.g., UV Roll, Move Z) are powerful photogrammetric tools and use a standard notation.

Given multiple views of a scene, it is often useful to represent the camera directly on the screen as an object. This screen representation must be explicitly invoked by choosing Camera from the **Create Object** menu, then specifying the source and destination views. The lower right pane of Figure 7 shows the two cameras (Overhead and Side-Looking) as seen from a third view. Seeing the camera represented in another view is quite useful when moving the camera around, especially when the user is not comfortable with using the perspective camera model.

When the camera rotations are complete, the three panes look exactly as seen in Figure 7. The upper right pane contains a view of the farm scene, with a house and silo in the foreground and a quonset hut in the background. The lower left pane is a view of the same farm from down the road. Finally, the lower right pane shows the scene from farther away at ground level; the two pyramid-like objects on the top and left are the cameras whose views are shown in the first two panes. The camera rays converge at the camera's focal point, and the image plane is visible. CME's sophisticated support of 3-D transformations allows multiple views to coexist in a smoothly integrated fashion.

This scenario shows how to create and manipulate a scene of 3-D objects, which can be viewed by multiple cameras from any perspective. The cameras can also be viewed as objects themselves, aiding the user in his interpretation of the scene. Although ImagCalc is used as a basis for the user interface, this scenario illustrates the flexibility of the CME Object system.

6 Scene Rendering

The separate capabilities of both ImagCalc and Object can be used together to form an even more powerful exploitation environment. This scenario illustrates how to model a group of buildings and render them using two images. The result is a realistic 3-D model that can be used to generate fly-through image sequences. This scenario will not be discussed in explicit detail since its operations are considerably more complex than the others.

Figure 8 illustrates the process of building a wire frame model using aerial imagery. Two distinct images of the same scene are put into both of the upper and lower panes, respectively. Then a wire frame model of the important buildings is built using the techniques illustrated in the modeling scenario. A second view of the model is created and the second camera is adjusted so that the model matches the orientation of the buildings in the underlying image. Ideally, the image views are different enough to permit a good fit of the model to the images, although some adjustment of individual buildings may be required.

After the model construction is completed, the model faces are rendered with photo textures extracted from one of the images. The CME **Render** operations are used to assign texture to the model. The result, illustrated in Figure 9, is a 3-D model with realistic texture and shading. Since the scene is three-dimensional, it can be viewed from various angles to support applications such as fly-through mission rehearsal.

In summary, this scenario illustrates the power of CME to integrate its component programs in a seamless fashion. An intuitive, powerful interface aids the user generating realistic, three-dimensional scenes from two-dimensional imagery.

7 Software Design

CME is a powerful environment for developing new IU algorithms. Some of the features which make CME such an environment are described below.

Object-Oriented Design CME is an object oriented design and is currently based on the Symbolics Flavor System. A portion of the object hierarchy is shown in figure 10. The hierarchy is based on object inheritance, where objects near the top of the hierarchy are more general than those at the bottom. A good example of the use of inheritance is the **vector-image** class. The **vector-image** defines a set of **image** layers. Most of the image processing



Figure 2: ImagCalc Zooming

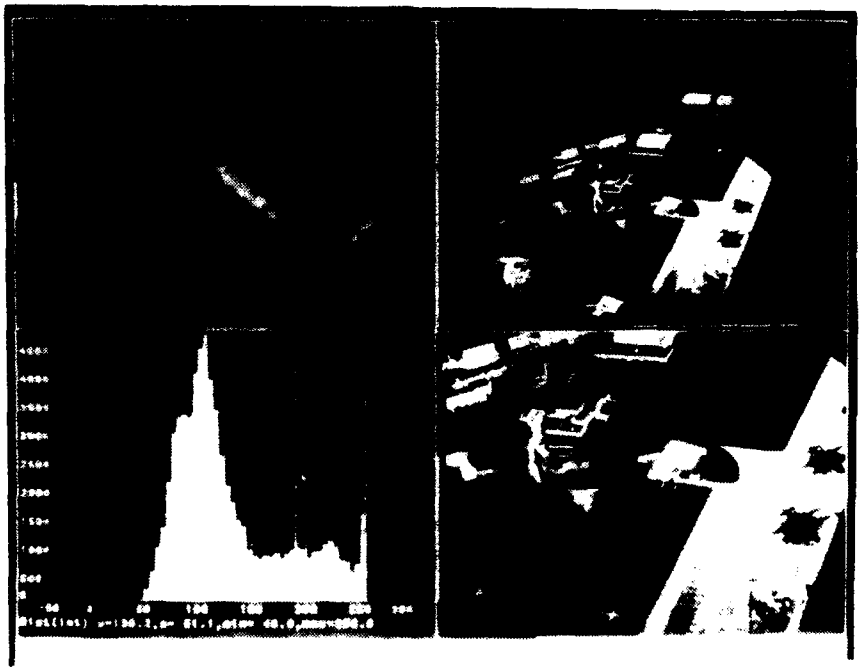


Figure 3: ImagCalc Contrast Stretching

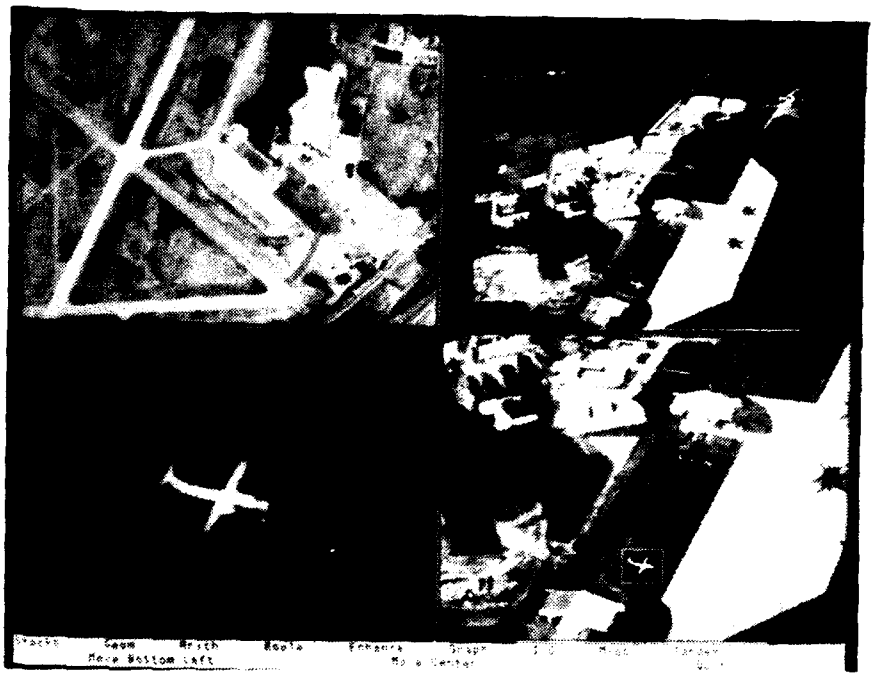


Figure 4: ImagCalc Windowing

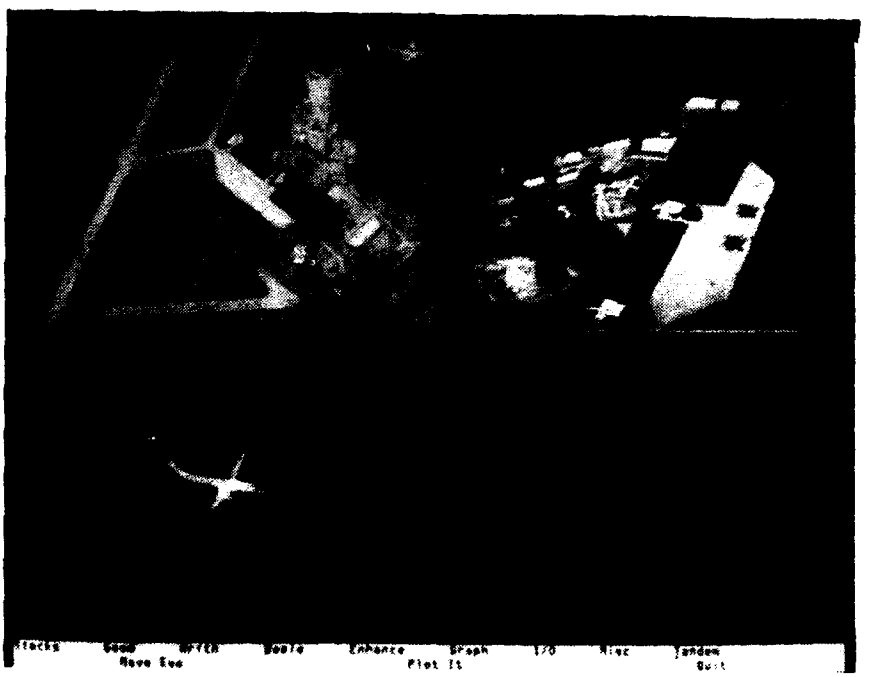


Figure 5: ImagCalc Depth Mapping

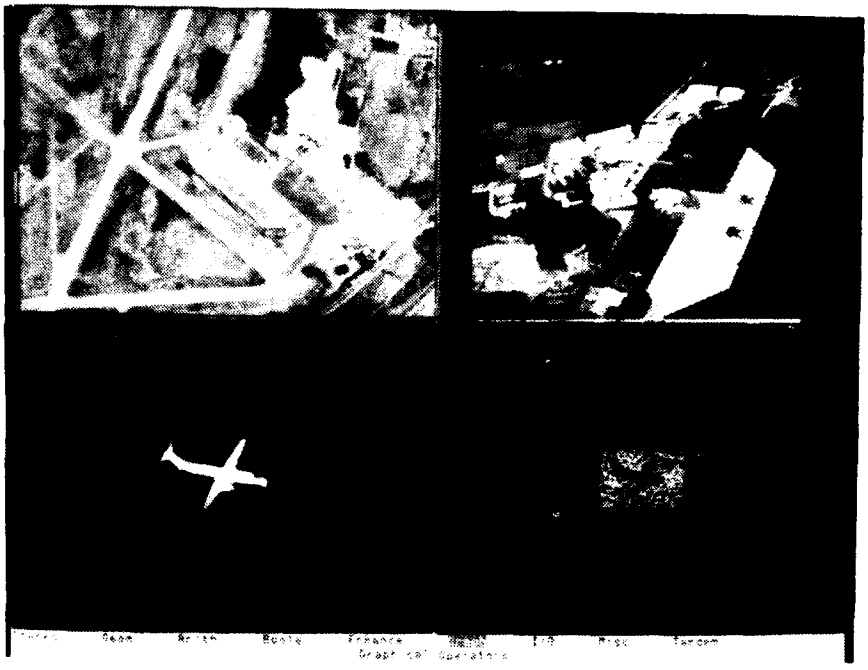


Figure 6: ImagCalc Edge Detection

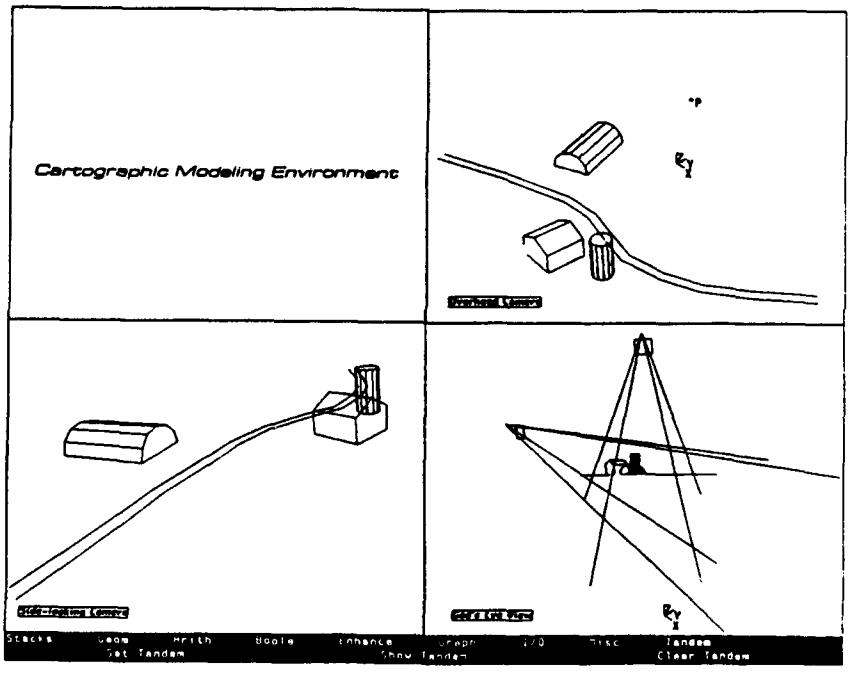


Figure 7: Object Scene Modeling



Figure 8: Modeled Image Scene



Figure 9: Rendered Object Model

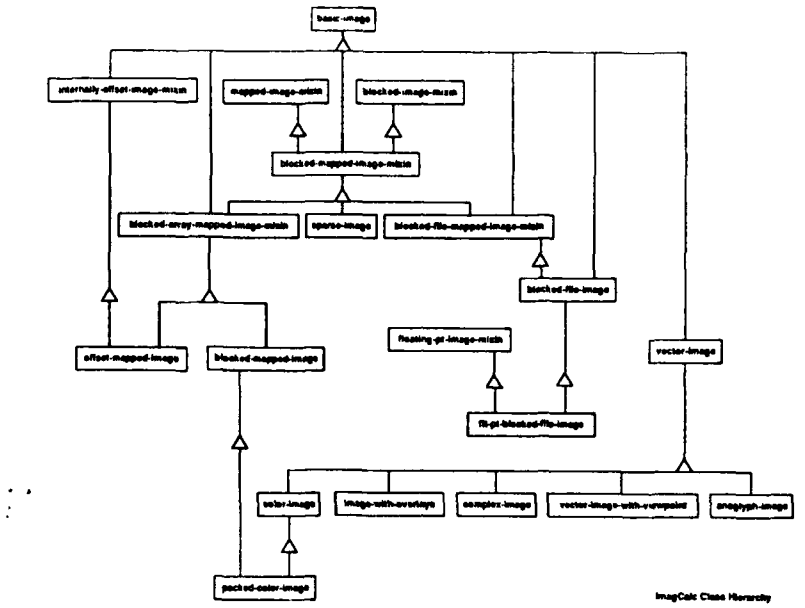


Figure 10: The portion of the object hierarchy for images. The power of inheritance is quite clear in the case of vector images where a RGB color image is considered as a special case of a set of image layers. The complex image produced by the FFT is another subclass of the vector image.

operations which apply to an **image** also apply to the **vector-image**, such as :zoom, :histogram and :warp. The programmer can proceed without considering the existence of multiple layers in many cases. Some special cases of the class **vector-image** are **color-image** and **complex-image**. Many of the operations for these more specific classes are inherited from the **image** component class or inherited from the general **vector-image** as a natural consequence of the object-oriented language.

Macros CME also makes extensive use of the Lisp macro facility to provide an efficient and extensible programming environment. Especially powerful, are CME's use of iteration macros so that functions which operate on all the pixels of an image or the vertices of a polyhedron can be written without direct coding by the programmer. For instance the macro **image-point-operator** allows one to map the same n-ary function at each point of an image set. This function is optimized for maximum speed using techniques which the average programmer would not be familiar with. For example, a restricted version of the **negate-image** function, designed to operate only on images containing 8-bit pixels, could have been defined as

```
(defun negate-image (image)
  (image-point-operator ((image))
    (lambda (x)
      '(- 255 ,x))))
```

using **image-point-operator**, while a simplified version of a function to add two 8-bit images would be

```
(defun add-images (image-a image-b)
  (image-point-operator ((image-a image-b))
    (lambda (a b)
      '(ash (+ ,a ,b) -1))))
```

Another useful macro is **with-objects-desensitized**. Normally all three dimensional objects in the CME system are mouse sensitive, meaning that they respond to the mouse by changing color and changing the meanings of mouse clicks. If a programmer desired to temporarily defeat that feature, for instance to allow the mouse to become sensitive to same graphical overly created independently of the three-dimensional object system, he could use macro in code such as the following:

```
(defun frob-my-graphics (graphic-element)
  (with-objects-desensitized
    (sensitize graphic-element)
    (frob-my-graphics-internal graphic-element)
; Do whatever
    (desensitize graphic-element)))
```

Process History Another interesting design feature of CME is the cached function history mechanism. The relational network is set up with data types as nodes and functions as edges. For example, suppose a the function, :threshold, is applied to image, I_1, which produces the 1-bit image, I_2. The resulting network is $[I_1 \leftarrow \{: threshold\} \leftarrow I_2]$ The history network provides two important services.

- If a data item already exists in the history network it is not necessary to recompute it. For example, suppose an image is bi-linearly interpolated to produce a 2x higher resolution image. Then later the operator decides to apply a 2x zoom-out reduction. Instead of applying another bi-linear mapping to the high-resolution image, CME determines that the desired result is already in the history and simply returns the previous image. This mechanism improves the efficiency of interaction for operations which require a significant amount of computation.

- Another application of the history mechanism is to record the sequence of operations and the parameter settings which are involved in computing a final result. For example, constructing an image segmentation requires on the order of a dozen operations including various convolutions and thresholding operations. It may be the case that the operator has produced the result by applying a series of menu operations. The history mechanism keeps track of the operations and the data items involved and produces a corresponding Lisp expression. The expression is in a form which can be easily edited to produce a general function for repeating the complex process. The result is a combination of the best features of a graphical user and an interpreted language interface.

8 The Rehost of CME

While the Symbolics Lisp Machine provides an effective environment for rapid prototyping and for carrying out experiments in image exploitation, it is clear that in the future most image understanding (IU) laboratories will be equipped with Unix workstations. It is also the case that many IU labs are currently developing their programs in the C or C++ language. Consequently, CME is currently being rehosted to run in Common Lisp on Unix platforms and an interface is being developed to provide C and C++ access to CME functions.

The rehost process has raised many interesting issues both in language interfaces as well as the problem of achieving high performance graphics and image processing functions in an X-window, Unix environment.

9 Conclusion

In summary, the RCDE is an exciting experiment in software engineering which should provide a new approach to the integration of image understanding algorithms.

SECTION III

Future IU Applications

Understanding Synthetic Aperture Radar Images

Rama Chellappa
Department of Electrical Engineering
Center for Automation Research
UMIACS*
University of Maryland
College Park, MD 20742

Ed Zelnio
WL/AARA
WP-AFB
Ohio 45433-3050

Eric Rignot
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

Abstract

Although over twenty years of basic and applied research work is available for understanding images in the visible spectrum, computational vision algorithms developed for application to visual imaging are usually not directly applicable to radar imagery because of the fundamental differences between the physics of synthetic aperture radar image formation and that of conventional images at visible wavelengths. Yet many of the paradigms that have evolved in image understanding (IU) and Computer Vision (CV) research are suitable for radar imagery. In this paper we summarize some of the existing IU and CV approaches to understanding of SAR images and discuss possible future research directions. Specific attention is given to problems such as segmentation, terrain estimation, change detection and target detection.

1 Introduction

Algorithms developed for application to visual imagery are usually not directly applicable to radar imagery. However, many of the approaches that have evolved in IU and CV research are suitable for radar imagery. Translation of paradigms and methodologies into algorithms for a specific application requires a model, either explicitly in a formal derivation or implicitly in the experiences of the algorithm designer. Models used in IU and CV are inappropriate for SAR imaging because of the fundamental differences between the physics of SAR image formation and that of conventional images at visible wavelengths. The purpose of this paper is to help bridge the gap between IU and SAR researchers by giving several examples of the applicability of some of the existing IU methods for SAR imagery. Earlier attempts at transferring IU methodology for the interpretation of SAR imagery are documented in [63].

We begin with a discussion of different types of SAR data that are available. Two models for explaining the SAR data are then presented. The first model uses the physics of the SAR imaging and processing system to characterize the statistics of speckle. Then the observed complex, multilook intensity, multifrequency and multipolarimetric SAR data are modeled using appropriate probability distributions. The second model relates the surface elevation, the nature of the surface, the viewing angle and the observed intensity. Such a computational vision model is useful when one is interested in estimating terrain elevation from the given SAR intensity data by using an algorithm modeled after existing shape from shading algorithms.

Several examples illustrating the applicability of IU paradigms are then given. Specifically, we discuss preprocessing, segmentation of single-look single-frequency and multi-frequency complex data, terrain estimation, and object recognition. Several possible future research efforts are also outlined.

2 Background Information

2.1 Physics Based Models for SAR Complex Data

In this section the basics of the SAR system, configuration, imaging process, and digital post-processing operations are presented. More details are also available in numerous valuable references and textbooks [17, 70, 73, 76, 75, 74]. In addition, the types and specifics of SAR data generated by current SAR systems and processors are presented to help the reader follow the adaptation of the segmentation technique to different types of SAR data.

2.2 SAR imaging and processing systems

SAR imaging. Figure 1 shows the typical imaging configuration of a strip map SAR instrument carried aboard an aircraft flying at an altitude H, along a straight line called the along track or azimuth direction, at a constant ground speed v, and imaging a surface strip S.

The cross track direction of imaging is referred to as the slant-range direction. The antenna beam is pointed to the side of the aircraft making an angle (look angle) with the vertical direction z. The local incidence angle i measures the angle between the local slant-range direction r and the local vertical direction z. The ground-range axis corresponds to the projection of the line of sight of the radar (slant-range) to the ground.

Imaging in the range direction results from accurate time-delay measurements of echoes received from many

*University of Maryland Institute for Advanced Computer Studies

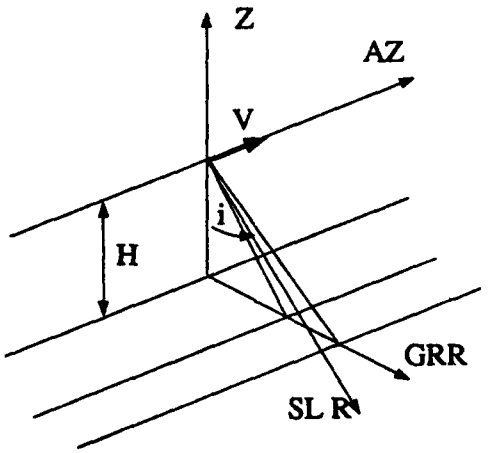


Figure 1: Imaging configuration of a strip map SAR instrument.

successive coded pulses. The achievable slant-range resolution R_r is determined by the smallest measurable time difference between the received echoes from two closely spaced targets on the same range line. If the coded pulse is a chirp (as is often the case), the slant range resolution of the instrument is

$$R_r = \frac{c}{2B_c} \tag{1}$$

where B_c is the chirp bandwidth ($B_c = 1/\tau$ where τ is the pulse duration), and c is the speed of light. The amplitude and phase of the returned electromagnetic signal are measured for each resolution element in slant range.

As the individual targets "move" through the antenna footprint, the phase of the corresponding echoes is Doppler shifted due to the variation in the traveling distance of the signal. In a real aperture radar, the phase information is not utilized and azimuth resolution is limited by the size of the antenna footprint. In a synthetic aperture radar, fine resolution imaging in the azimuth direction is achieved by recording the Doppler phase history in azimuth of each target so that the sampled echoes can then be added on a coherent basis. The process is equivalent to the use of an array of antennas or to a synthetic larger antenna. Spatial resolution in azimuth, defined by the size of the synthetic antenna footprint, is

$$R_z = \frac{L}{2} \tag{2}$$

where l is the physical length of the antenna along the azimuth. Azimuth resolution does not depend on the altitude of the flying craft. As range resolution is also independent of the altitude, SAR instruments permit high resolution imaging from space with antennas reasonably small in size. In addition, in the microwave region, as the atmosphere of the planet is nearly transparent, SAR imaging is possible in all weather conditions. Imaging is also independent of sun illumination as the SAR provides its own source of illumination of the surface.

SAR Processing. Processing of the SAR raw data consists of range compression where the returned echoes are cross-correlated with a replica of the transmitted pulses, and azimuth compression which is a similar correlation process, but range dependent, performed in the azimuth direction. Correlation is traditionally performed in the frequency domain via Fourier transforms. Alternate forms of SAR processing are also possible [17]. Different scanning modes for the SAR (burst mode, spotlight mode, scan mode) require different processing strategies.

Digital SAR processing can be computationally involved. The data must be corrected from the altitude and trajectory of the craft carrying the instrument, the plane rotation, the antenna pattern illumination, possible ambiguities due to sampling of the signal, and errors in focussing (compensation of the signal for the Doppler shift during generation of the synthetic array). Numerous filtering techniques are used to control computer storage and arithmetic complexity, and enhance the image quality. In the case of multifrequency and polarimetric radar systems, data transmission, reception, handling, processing, calibration, and storage are also much more demanding.

SAR post-processing operations. A number of post-processing operations are performed to facilitate the manipulation and use of the data for scientific purposes. The most important one, often already included in the SAR processor itself, is multilooking. Multilooking is an artificial way of reducing image speckle in SAR imagery and improving the radiometric resolution of the data by incoherently averaging multiple independent SAR data samples of the same scene. Independent looks are generated from a single data take by processing different parts of the available SAR signal bandwidth [61]. Each part of the spectrum produces an image, and the resulting images are incoherently added to produce one multilook image. Image speckle and the data volume are reduced, two important advantages for large scale and high data rate analysis of SAR data with simple algorithms. The drawbacks are a reduction in spatial resolution by a factor proportional to the number of looks, loss of absolute phase information, and degradation of the analytical tractability of the statistics of the data.

Because of its viewing geometry, a SAR collects data in the slant-range direction. To obtain information in the ground-range direction (better suited for multisensor coregistration and synergistic utilization of the data [48, 66]) the image must be corrected from a nonlinear stretch in the range dimension and a skew in the azimuth direction [18, 19]. This transformation is accompanied by a resampling of the data which modifies the spatial statistics, an important aspect to consider in

the development of automated techniques of analysis of SAR data.

SAR data are also radiometrically rectified and calibrated. Rectification accounts for the antenna pattern of the illumination and errors in antenna pointing. Calibration assures the short term relative (within image), long term relative (from one image to another), and absolute (independent of the instrument) radiometric fidelities of the radar measurements. In the case of polarimetric data, leakage between the different channels must also be accounted for [28, 79].

SAR noise sources and distortions. Thermal noise from the receiver electronics is not the only source of noise in SAR. Quantization noise, bit error rate noise, ambiguity noise and side-lobe noise are additional noise sources in the signal. A convenient and common way of characterizing the total system noise power level is to determine its noise equivalent backscatter cross-section ($NE\sigma_0$). Typically, the $NE\sigma_0$ is -40 dB in airborne SAR and -20 dB in spaceborne SAR, whereas the backscatter cross-section of natural targets lies in the range $+5$ dB to -35 dB.

In most conventional passive sensors, the dominant noise source is system noise. In a coherent imaging sensor such as SAR, the dominant source of signal fluctuations is image speckle. Signal fluctuations due to system noise are much lower in amplitude except in the case where the backscatter cross-section of the target reaches the $NE\sigma_0$ floor. Image speckle results from the constructive and destructive interferences of multiple returns from individual scatters within the same resolution cell [33]. In digital data, image speckle acts as a multiplicative noise on the radar signal. In single look SAR data, the mean square to variance ratio of the intensity of the radar signal is 1, i.e. the inherent signal-to-noise ratio is 0 dB.

Yet, there is a certain danger in referring to image speckle as a noise source, although this is widely done in the SAR literature. In terms of its physics, image speckle is not noise but information. It is in fact the principal source of information used in SAR interferometry [32], i.e. SAR interferometry would be impossible if image speckle were systematically removed from the data. Depending on the application, one must always carefully examine whether image speckle should be treated as perturbation or information.

An additional difficulty in the interpretability of SAR imagery is the geometric distortions resulting from the viewing geometry of radars. In the presence of terrain, effects such as layover, shortening and overshadowing [51] distort the geometry and complicate the inference of terrain height from the data [26, 27].

2.3 Types and specifics of SAR data

Using quadrature filters the voltage measured at the receiving antenna of a SAR is decomposed into both a magnitude $|\,\mathbf{a}\,|$ and a phase of the returned electromagnetic signal. The corresponding measurement is represented as a complex number called the complex amplitude of the signal

$$\mathbf{a} = |\,\mathbf{a}\,| \exp\{-\mathrm{j}\phi\} \tag{3}$$

and the square magnitude of this complex amplitude is called the intensity (or power) of the radar return

$$I = |\,\mathbf{a}\,|^2 \tag{4}$$

In addition, a SAR instrument can operate (sometimes simultaneously) at several frequencies and polarization configurations of the antenna, and provide multitemporal imaging of the same scene. Multilooking may be performed on these multidimensional measurements. These different imaging modes generate different types of data described next.

Single frequency, single polarization SAR data can be of two types: 1) SAR complex data: 2) multilook SAR intensity data.

SAR single look complex data. Except for the case of SAR interferometric studies, no coherent spatial averaging is performed on these data as it does not improve radiometric resolution and degrades spatial resolution. SAR complex data therefore correspond to the simplest form of processed information from the radar and also to the best spatial resolution available from the SAR imaging and processing systems. On the other hand, image speckle dominates the statistics of the data. At the pixel level, it is characterized by a low correlation coefficient (typically less than two pixels), and a variance to mean square ratio of 1. A data display of the magnitude of the amplitude appears noisy with speckle patterns, and the absolute phase of the signal is nearly uniformly distributed and difficult to analyze.

Multilook (N-look) intensity data. After detection of the complex signal, a single look intensity image of the measurements is obtained. Single look intensity images are not adapted for visual interpretation of the SAR data because of image speckle. Instead, multilook intensity is the most common SAR data format used by the research community as it is practical for visual interpretation and analysis of large areas with simple algorithms.

Polarimetric SAR complex data. The polarization state of an electromagnetic wave describes the relative motion of the vector representing the electrical field when the wave moves towards an observer. Conventional radars use the same antenna polarization configuration for both transmission and reception of the electromagnetic signal and operate at a fixed polarization state. In recent years, fully polarimetric radars have been developed that decompose the signal into two orthogonal signals, one horizontal and the other vertical (Figure 2), that are received and processed independently in separate channels. By combining several horizontal/vertical polarization configurations at both the transmission and reception ends of the imaging system the complete polarimetric complex scattering matrix of each resolution element of the image is measured, i.e.

$$\mathbf{S} = \begin{pmatrix} S_{HH} & S_{HV} \\ S_{VH} & S_{VV} \end{pmatrix} \tag{5}$$

where S_{XY} is the complex amplitude of the signal in the Y polarization when the transmitted electromagnetic wave is X polarized, and X and $Y \in \{H, V\}$. The recorded complex scattering matrix of each element can

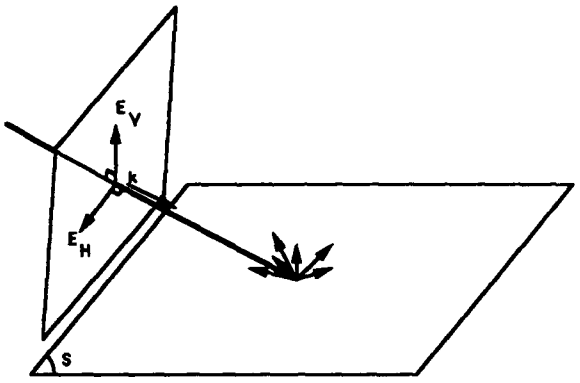


Figure 2: Definition of horizontal (H) and vertical (V) in a polarimetric image.

in turn be used to synthesize any type of polarization configuration L of the radar antenna using

$$\mathbf{S}_L = b_r^{*T} \mathbf{S} b_t \tag{6}$$

where T denotes transposition, and $b_{t,L}$ and $b_{r,L}$ are the wave orientation matrices of, respectively, the transmitted wave and the received wave for configuration L [87]. To generate an $L = HH$ complex amplitude,

$$b_t = b_r = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \tag{7}$$

For a circularly right polarized return,

$$b_t = b_r = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \tag{8}$$

Single look polarimetric complex data. The scattering matrix of each sample element of the image is stored in a compressed format [78]. These data are required to perform the accurate radiometric multichannel calibration of the data, and correspond to the best spatial resolution available from the SAR imaging and processing systems.

Multilook polarimetric SAR complex data. Multilook averaging of polarimetric SAR data is performed as follows:

$$\langle S_{XY} S_{XY}^* \rangle = \sum_{i=1}^{i=N} S_{XYi} S_{XYi}^*$$

$$\langle S_{XX} S_{XY}^* \rangle = \sum_{i=1}^{i=N} S_{XXi} S_{XYi}^* \tag{9}$$

i.e. it corresponds to the spatial averaging of the scattering matrices. The data are still complex amplitudes but the absolute phase information is lost, and only the relative phase difference between the different polarimetric channels is preserved. The effect on the intensity ($\mid S_{XX} \mid^2$) is identical to a multilooking operation performed in the time domain. The cross product coefficients ($\langle S_{XX} S_{XY}^* \rangle$) are unchanged, i.e. the mean phase difference is preserved as well as the correlation coefficient between the different channels. On the other hand, their variance is reduced by a factor proportional to the number of looks. Multilook polarimetric SAR complex data are stored in a standardized compressed format which permits one to retrieve the multilook amplitude and the multilook phase difference of the complex scattering coefficients [22].

Multifrequency SAR data. The frequency of the radar electromagnetic wave is a key factor in the penetration depth of the signal. As the wavelength increases, the signal penetrates deeper into the ground, and images different layers of the surface and near subsurface which may be of different physical structure, dielectric constant, or surface roughness. Frequency is also a key factor in scattering from a rough surface. In the Bragg model, the backscatter cross-section of a rough surface is proportional to the roughness spectrum of the surface and the fourth power of frequency. Numerous examples have already shown [76] that multifrequency information helps separate different types of natural surfaces and characterize their physical and electrical properties better than single frequency data.

The electromagnetic radar signal is also characterized by its frequency which, by convention, is designated by a letter of the alphabet. For instance, P- band ranges from 225 to 390 GHz, L- band from 390 to 1550 GHz, C- band from 3900 to 6200 GHz, and X- band from 6200 to 10900 GHz. Several airborne SAR instruments are currently capable of near simultaneous acquisition of SAR data at several frequencies. The SIR-C experiment will be the first of its kind to operate a multifrequency SAR from space, and the upcoming EOS SAR platform is planned to operate at L-, C- and X-bands.

2.4 Model of the speckled complex amplitudes

Joint distribution of complex amplitudes. Let us assume that speckle is fully developed as in [33] , i.e.:

- The phase ϕ and the magnitude $\mid a \mid$ of the signal are independent random variables.
- The contribution of each element to the total return is independent of the other scattering elements contained in the resolution cell.
- The phase ϕ of the signal is uniformly distributed.
- The number of scattering elements per resolution cell is much larger than one.

Under these assumptions, the complex amplitude of the signal has circular complex Gaussian statistics

$$p(a) = \frac{1}{\pi} \exp\{-\frac{\mid \mathbf{a} \mid^2}{< I >} - \log < I >\} \tag{10}$$

where $< I >$ is the local mean intensity of the signal. The complex amplitudes are zero mean.

To include contextual information from neighboring pixels and improve the segmentation process, the

joint distribution of M complex amplitudes $\mathbf{a_s} = [a_1, \ldots, a_M]_s^T$ in a neighborhood N_s of pixel site s is examined instead of the distribution of a single complex amplitude at pixel site s. The segmentation technique can then exploit the additional information provided by the neighbors of s to improve the segmentation.

The complex image array Ω is viewed as a set of $P \times Q$ (where P and Q are the numbers of pixels in the vertical and horizontal directions) conditionally independent overlapping windows of M elements each, centered at each pixel site S, such that each pixel within that window has the same local mean intensity level $< I >$. The joint distribution of the M complex amplitudes is

$$p(\mathbf{a_s}) = \frac{1}{\pi^M} \exp\{-\mathbf{a_s}^{*T} \mathbf{R}_{\mathbf{a_s}}^{-1} \mathbf{a_s} - \ln | \mathbf{R_{as}} |\} \quad (11)$$

$\mathbf{R_{as}}$ is the $M \times M$ complex matrix of the M zero-mean complex amplitudes $\mathbf{a}_s$ at site s (i.e., $\mathbf{R_{as}} =< \mathbf{a_s a}^{*T} >_s$, where $<>$ denotes ensemble averaging), and $| \mathbf{R_{as}} |$ denotes the determinant.

This model is in principle not applicable when speckle is no longer fully developed. Such is the case in areas where a single scatter dominates, the number of scatters per resolution is small, or the scale of the surface roughness is no longer much smaller than the wavelength of the electromagnetic signal.

This model also assumes that the backscatter cross-section of the imaged surface is uniform. When this is not the case, (11) must be convolved with the distribution of the backscatter cross-section of the surface. Because of the presence of image speckle, this distribution is not directly observable and is difficult to infer. A gamma distribution of the backscatter cross-section results in a K-distribution of the speckled complex amplitudes and has been shown in several applications to better describe the statistics of radar measurements [40]. The variability of the backscatter cross-section of the surface, called texture, is quantified using a single parameter α which measures the increase in the variance of the signal compared to that due to image speckle alone. On the other hand, texture has no effect on the correlation properties of the complex data.

Correlation matrix of the complex amplitudes. Let us consider the SAR imaging and processing system as a coherent system which transforms a complex amplitude input into a complex amplitude output.

Earlier studies [2, 55, 85] showed that the coherent system can be modeled as a linear shift invariant system characterized by its two-dimensional impulse response function h provided that the across-track dimension is slant-range (as opposed to the projected ground-range direction for which h would be shift variant) and the along-track direction is azimuth (Figure 1). As a consequence, for homogeneous regions,

$$\mathbf{R_{as}} =< I >_s R_h, \quad (12)$$

where R_h is the correlation matrix of the system coherent impulse response h, and $< I >_s$ the ensemble average of the intensity of the complex data at pixel site s. Hence, the modeling of $R_{\mathbf{as}}$ is directly related to the SAR system coherent impulse response function.

Ideal response from a point target. We assume for the moment that SAR processing is comprised of two independent operations: range compression and azimuth compression. In digital imagery, as pixel elements are manipulated instead of resolution elements, the components of the correlation matrix of the ideal coherent system impulse response are rewritten in terms of pixel spacing as

$$R_h(m,n) = \mathrm{sinc}(m\frac{s_r}{R_r}) \mathrm{sinc}(n\frac{s_z}{R_z}), \quad (13)$$

where, respectively, m and n are slant-range and azimuth pixel indexes, s_r and R_r are the sample spacing and spatial resolution in slant-range, and s_z and R_z are the same quantities in azimuth. Note that spatial resolution here is not equivalent to the conventional 3 dB bandwidth of the impulse response r_r; rather,

$$r_r = 0.8845 R_r \quad (14)$$

The correlation matrix R_h given in (13) has several remarkable properties that have been pointed out independently by various authors [55, 59, 64].

- R_h is also equal to the correlation function of speckle as the bandwidth of speckle is determined by the width of the synthetic aperture.
- R_h is not affected by the presence of texture, i.e. the spatial coherence of the imaged surface.
- R_h is insensitive to focussing errors.

In general, the system response of a sensing instrument is difficult to estimate or characterize. SAR complex data provide a unique opportunity where the system coherent impulse response function can actually be accurately characterized, measured and modeled.

2.5 Model of the multifrequency speckled complex amplitudes

Speckle is assumed to be fully developed as in Section 2.4. The complex amplitude of the signal is circular Gaussian and the joint probability distribution function of a set of M neighboring complex amplitudes $\mathbf{a_1} = [a_{1,1}, \ldots, a_{1,M}]$ at a frequency f_1 is

$$p(\mathbf{a_1}) = \frac{1}{\pi^M} \exp\{-\mathbf{a_1^{*T}} \mathbf{R_1} \mathbf{a_1} - \ln | \mathbf{R_1} |\} \quad (15)$$

where $\mathbf{R_1}$ is the correlation matrix of the complex amplitudes at frequency f_1.

If the same scene is imaged at a different frequency f_2, the resulting complex amplitudes are mathematically independent. The reason is that the radar signal is sensitive to scatterers of a different nature (different roughness scale, dielectric constant, etc.) and that consequently image speckle is uncorrelated. The complex amplitudes are therefore uncorrelated, and since they are circular Gaussian they are also independent. The joint distribution of M complex amplitudes at two frequencies is therefore the product of the joint distributions at each frequency,

$$p(\mathbf{a_1}, \mathbf{a_2}) = \frac{1}{\pi^{2M}} \exp\{-\mathbf{a_1}^{*T} \mathbf{R_1^{-1}} \mathbf{a_1} - \ln | \mathbf{R_1} | - \mathbf{a_2}^{*T} \mathbf{R_2^{-1}} \mathbf{a_2} - \ln | \mathbf{R_2} |\} \quad (16)$$

The correlation matrices $\mathbf{R_{i,l}}$ are related to the correlation matrices of the system coherent impulse responses R_h^i as in (17),

$$\mathbf{R_{i,l}} =< I_{i,l} > R_h^i \tag{17}$$

where $< I_{i,l} >$ is the mean intensity of the signal at frequency f_i in region l. As spatial resolution is frequency dependent, R_h^i depends on frequency. Most current multifrequency SAR imaging and processing systems are designed so that spatial resolution is nearly constant across frequency to facilitate the coregistration and multidimensional analysis of the multifrequency measurements. R_h^i has the same properties as those encountered in Section 2.4 and is estimated using the same technique.

2.6 Model of multilook SAR intensity data

The marginal distribution of multilook SAR intensity data is not accurately known. When the samples used to generate the multilook data are independent (which may not be the case), and the backscatter coefficient of the surface is locally uniform, the multilook intensities are gamma distributed, i.e.

$$p(I_i) = \frac{N^N I_i^{N-1}}{< I >^N \Gamma(N)} \exp\left(-\frac{N I_i}{< I >}\right) \tag{18}$$

where $< I >$ is the local mean intensity of the signal and N is the equivalent number of looks, here equal to the number of independent single look samples. If the backscatter coefficient of the surface is not uniform but gamma distributed with parameter α, the intensity is K-distributed [40].

$$p(I_i) = \frac{2\alpha}{\Gamma(\alpha)\langle I\rangle}(\alpha I/\langle I\rangle)^{\frac{\alpha-1}{2}} K_{\alpha-1}(2\sqrt{\alpha I/\langle I\rangle}) \tag{19}$$

where $K_{\alpha-1}$ is the modified Bessel function of the third kind of order $\alpha - 1$.

If the samples are correlated, both (18) and (19) are not mathematically correct. The exact form of the distribution of the multilook SAR intensities, discussed in [33], is complicated. A computationally convenient alternative is to adapt the equivalent number of looks N in (18) or the parameter α in (19) to correctly model the larger than expected variance of the signal. Although K-distributions seem to better describe the statistics of SAR data in several applications, the rather empirical gamma distributions have been shown to provide a reasonable model of the statistics of SAR data.

The correlation properties of SAR intensity data produced by a partially coherent system which transforms a complex amplitude input into an intensity output have been studied by several authors [55, 59, 64]. As focussing, texture, and multilooking have a significant impact on the shape of the system incoherent impulse response, the results are different from those obtained in the case of a completely coherent system.

If the scene has stationary statistics and the system is perfectly focussed, the correlation function of single look intensity data ($\mathbf{R_I}(\mathbf{b}fr) =< \mathbf{I}(\mathbf{U})\mathbf{I}(\mathbf{u}+\mathbf{r}) >$) is given by the Siegert relation

$$\mathbf{R_1}(\mathbf{r}) =< I >^2 + < I >^2 |\, R_h(\mathbf{r})\,|^2$$

where $\mathbf{r} = (m, n)$ is the displacement vector. If texture is present, the correlation function becomes [59]

$$\begin{aligned}\mathbf{R_I}(\mathbf{r}) =< I >^2 \;&+\; < I >^2 |\, R_h(\mathbf{r})\,|^2 \\ &+\; \int\int [|\, h(\mathbf{u})\,|^2 |\, h(\mathbf{v})\,|^2 \\ &+\; h(\mathbf{u})h^*(\mathbf{v})h^*(\mathbf{b}-\mathbf{r})]\mathbf{K}_{<\mathbf{I}>} \\ &\quad (\mathbf{u}-\mathbf{v}+\mathbf{r})d\mathbf{u}d\mathbf{v}\end{aligned} \tag{20}$$

where $\mathbf{K}_{<\mathbf{I}>}$, the covariance function of the local mean intensity, measures the spatial variability of the backscatter cross-section of the surface. The additional component is composed of an incoherent term that corresponds to the incoherent imaging of the scene backscatter fluctuations and of a coherent term similar to the correlation function of the coherent impulse response. The system incoherent impulse response function is affected by focusing [64]. If the SAR processor is out of focus, the system incoherent impulse is broadened, resulting in a blurred intensity image, and (20) is no longer correct.

Assuming that the covariance matrix of the intensity data $\mathbf{R_{I-\langle I\rangle}}$ is known, the joint distribution of M neighboring single look intensities is

$$\begin{aligned}p(I_1,\ldots,I_M/L_s = l) \;=\; &|\mathbf{R^{-1}_{I-\langle I\rangle}}| \exp\{-\sum_l s_{ii} I_i \\ &+ \sum_{j>i} \log[I_o(s_{ij}\sqrt{I_i I_j}/\langle I\rangle]\}\end{aligned} \tag{21}$$

where I_o is the modified zero order Bessel function of the first kind, and $S = (s_{ij})_{(i,j)\in\Omega} = R^{-1}_{I-\langle I\rangle}$ (see [33] for the case of $M = 2$).

When the number of looks is larger than one, the correlation function of the intensity is modulated by the multilooking filters used to select the independent samples. When multilooking is performed in the time domain using neighboring samples in azimuth, the system incoherent response is nearly a $\sin_c$ function in azimuth. In the general case, an exact analytical expression of the correlation function of multilook is complicated [55]. There is little need for exact modeling of the system incoherent impulse response function in multilook SAR intensity data as the spatial correlation of the intensity that is due to speckle becomes rapidly negligible as the number of looks increases. Spatial correlation of SAR intensity data is therefore mostly due to the spatial correlation of the backscatter cross-section of the imaged surface. Even in the presence of highly non-homogeneous features the uncorrelatedness assumption is adequate.

The adoption of a simple model for the correlation function of multilook SAR intensities is also justified by several other considerations:

- In the context of coregistration and synergistic analysis of multisensor data, remapping of the multilook SAR intensity data onto a common ground range projection is a requirement [48, 66], and the process will be implemented on a number of future operational SAR processors. As this operation is highly non-linear and results in a shift-variant impulse response function, the analytical modeling of

the correlation function of multilook intensities becomes difficult and may eventually lead to overly complicated segmentation algorithms.

- Even in the case of SAR single look complex data, where neighboring complex samples are strongly correlated, it has been shown that the uncorrelated model may yield the same level of segmentation accuracy as the correlated model [65].

If the SAR complex data are uncorrelated, single look SAR intensity data are independent and so are the multilook SAR intensity data. If the SAR complex data are correlated, in the limit of an infinite number of of looks, the SAR intensity data are uncorrelated and Gaussian distributed (central limit theorem), and therefore independent. In the case of a finite number of looks, although it is not mathematically correct for gamma distributions, one can assume that the uncorrelated multilook intensity data are independent as well. The joint distribution of the multilook intensities is approximated by the product of the marginal distributions. In the case of Gamma marginal distributions, the joint distribution of M intensities within a neighborhood N_s of pixel site s is of the form

$$p(I_1, \ldots, I_M) = \exp\{-MU_1(I_1, \ldots, I_M)\} \qquad (22)$$

where the energy function U_1 is

$$\begin{aligned} U_1(I_1, \ldots, I_M) &= \frac{1}{M} \sum_{i=1}^{i=M} \left[\frac{N I_i}{\langle I \rangle} - (N-1) \log I_i \right] \\ &+ N \log < I_l > + \log \Gamma(N) \\ &- N \log N \end{aligned} \qquad (23)$$

Each region is characterized by a mean intensity value $\langle I_i \rangle$. The entire intensity data array is characterized by its equivalent number of looks N.

2.7 Model of polarimetric SAR complex data

To be able to completely characterize the polarization properties of the backscattered signals it is necessary to decompose the wave into two orthogonal polarized components and measure the returned like and cross-polarized signals. In practice, radars have been designed to be able to acquire almost simultaneously the HH, HV, VV and VH returns of the transmitted wave, where the first letter represents the polarization of the signal at the transmission and the second letter at the reception. With the help of these four quantities any type of polarization configuration can be simulated (elliptic, circular, ...) [87, 80] and the polarimetric properties of the targets are completely characterized.

X_s denotes the polarimetric measurement vector at site s, i.e. the vector of the complex measurements (i.e. amplitude and phase of the electromagnetic response) at site s,

$$X_s = [HH, HV, VV]_s, \qquad (24)$$

The cross-polarized return HV is the complex amplitude of the polarized V response given that the transmitted signal is H-polarized. The HH and VV amplitudes are the co-polarized terms. The VH responses is not present in (24) as it is symmetrized with the HV response during compression and calibration of the data based on the reciprocity principle. Since the polarimetric complex amplitudes are circular Gaussian, the conditional distribution of X_s is [43]

$$p(X_s) = \frac{1}{\pi^3 \mid C_l \mid} \exp\{-X_s^* C^{-1} X_s^T\}. \qquad (25)$$

The 3×3 complex matrix $C = < X^{*T} X >$ is the polarimetric covariance matrix of the data.

Evaluation of (25) at each pixel location involves prohibitive calculations of the Hermitian form $(X_s^* C^{-1} X_s^T)$. In the case of azimuthally symmetric targets (valid for a large variety of natural targets) the HV amplitude is uncorrelated with the HH and the VV complex amplitudes and the covariance matrix is simply

$$C = \sigma \begin{pmatrix} 1 & 0 & \rho\sqrt{\gamma} \\ 0 & \epsilon & 0 \\ \rho^*\sqrt{\gamma} & 0 & \gamma \end{pmatrix} \qquad (26)$$

where

$$\begin{aligned} \sigma &= \langle \mid HH \mid^2 \rangle; \\ \epsilon &= \frac{\langle \mid HV \mid^2 \rangle}{\langle \mid HH \mid^2 \rangle}; \\ \gamma &= \frac{\langle \mid VV \mid^2 \rangle}{\langle \mid HH \mid^2 \rangle}; \\ \rho &= \frac{\langle HHVV^* \rangle}{\sqrt{\langle \mid HH \mid^2 \rangle \langle \mid VV \mid^2 \rangle}}, \end{aligned} \qquad (27)$$

C can be inverted analytically, leading to

$$\begin{aligned} p(X_s) &= \tfrac{1}{\pi^3} \exp\{-\tfrac{(HH)^2}{\sigma(1-|\rho|^2)} - \tfrac{|HV|^2}{\sigma\epsilon} \\ &- \tfrac{|VV|^2}{\sigma\gamma(1-|\rho|^2)} + 2\epsilon\sqrt{\gamma} \mid HH \mid\mid VV \mid\mid \rho \mid \end{aligned}$$

$$\cos(\phi_{HHVV} - \phi_\rho) - \log[\sigma^3 \epsilon \gamma (1 - \mid \rho \mid^2)]\} \qquad (28)$$

where $\phi_{HHVV} = \phi_{HH} - \phi_{VV}$ and ϕ_{HH}, ϕ_{VV} and ϕ_ρ are the phase of HH, VV, and ρ, respectively. The evaluation of (28) now requires a much smaller number of operations.

A certain number of classification techniques have already been published. Some are based on the selection of simple features to perform classification of the pixels (to reduce the dimensionality of the data), some others use the fully polarimetric information. A review of these techniques and a comparison of the results is given in [53]. [43] derived an optimal Bayesian classifier using the fully polarimetric information. Refinements of the method have appeared in [53, 86]. The work in [80] significantly improved the earlier classification results in [43] by adaptively and iteratively estimating the priori probabilities needed for the implementation of discriminant function. Improvements were also obtained [86] by normalizing the vector $\mathbf{X}$ by the total received power, i.e.,

$$\mathbf{X}_N = \mathbf{X} / [|HH|^2 + |HV|^2 + |VV|^2]$$

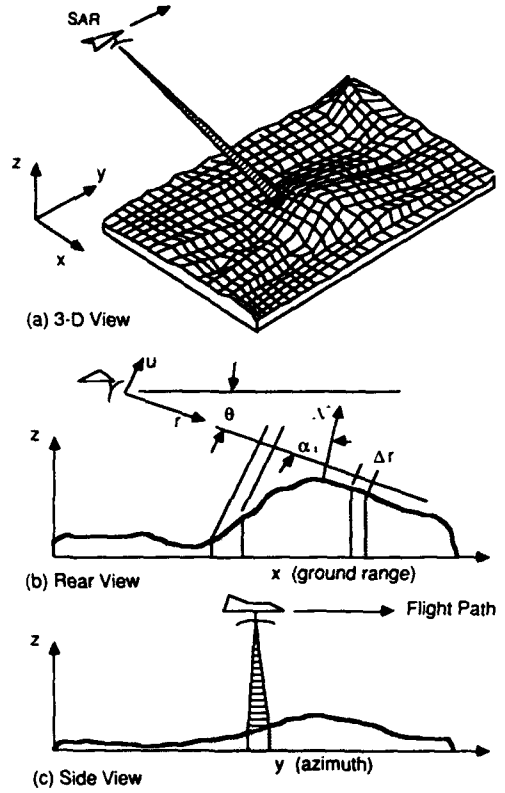


Figure 3: SAR Imaging Geometry.

2.8 Computational vision model

SAR Image Coordinate System

An image can be thought of as the projection of a 3-D scene into a 2-D representation. For an image created using conventional optics the spatial transformation from 3-D to 2-D is given by the perspective projection. The coordinate system for a SAR strip map is very different, as depicted in Figure 3. For simplicity, consider the case of a straight flight path with the antenna main beam orthogonal to the flight path (zero squint angle). Assume that the x-axis is lined up with the r-axis. Then x may be referred to as ground range and is related to slant range r by

$$r^2 = x^2 + (h - z)^2 \tag{29}$$

where h is the radar altitude and the origin is located on the ground directly below the radar. The second SAR strip map axis, by design, represents the along track distance y. Thus, SAR image coordinates are a projection of the physical scene, $(x, y, z(x, y))$, into the "slant plane", (r, y). One result is foreshortening of the mountainsides that slope upward as r increases.

Clearly the transformation from Cartesian coordinates, (x, y, z), to SAR image coordinates (r, y) is not an orthographic projection. However, it is possible to approximate the SAR coordinates as an orthographic projection of a rotated version of the surface. Suppose that r is large relative to the image size so that arcs of constant range are approximately straight lines over the depression angle subtended by the image. Then we obtain $u(r, y)$, the surface height relative to the slant plane defined by (r, y), as

$$\begin{pmatrix} r \\ y \\ u \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} r_o \\ y_o \\ u_o \end{pmatrix} \tag{30}$$

where (r_o, y_o, u_o) is an arbitrary reference point. Here the transformation from slant plane surface coordinates to SAR image coordinates is indeed an orthographic projection. In the absence of layover, $z(x, y)$ being single valued implies that $u(r, y)$ is single valued. To evaluate the reflectance map for surface reconstruction or image synthesis it is necessary to evaluate the angle of incidence α_i or its cosine. When expressed in (r, y, u) coordinates

$$\cos\alpha_i = \frac{u_r}{\sqrt{u_r^2 + u_y^2 + 1}} \tag{31}$$

where

$$u_r = \frac{\partial u}{\partial r} = \frac{\partial u_1}{\partial r} + \tan\theta \tag{32}$$

$$u_y = \frac{\partial u}{\partial y} = \frac{\partial u_1}{\partial y} \tag{33}$$

$$u_1(r, y) = u(r, y) - (r - r_o)\tan\theta \tag{34}$$

are the partial derivatives of u. For computer implementation, $u(r, y)$ is represented in a 2-D array with constant sample spacing in r and y as in a SAR strip map, and $u_1(r, y)$ is the deramped version of $u(r, y)$. Standard finite difference approximations are used for the partial derivatives.

This slant representation allows relatively simple and efficient processing techniques to be applied to SAR image synthesis from terrain evaluation data. Efficient surface reconstruction and image synthesis using the slant plane representation requires a method for resampling a digital terrain map (DTM) from ground coordinates to slant plane coordinates and vice versa. A method which is a variation of the summed area table method reported in the computer graphics literature [36] and is amenable to high speed implementation can be used.

The Reflectance Map

The reflectance map $R(.)$ is the function which relates the local surface orientation to image intensity. In both radiometry and photometry, $R(.)$ is a combination of the surface microstructure, the geometric properties of the illumination and the geometric properties of image formation. The reflectance model should be parameterized to retain the distinction between factors unique to SAR image formation and factors due to electromagnetic scattering properties of the surface while explicitly including the dependence on the surface slopes in both the range and azimuth directions. For SAR, the reflectance map $R(.)$ can be expressed as the product of an area factor A multiplied by the radar cross-section (RCS) σ_o per unit area. Both A and σ_o are functions of imaging geometry and surface orientation, i.e.

$$R(u_r, u_y, \beta) = A(u_r, u_y, \beta)\sigma_o(u_r, u_y, \beta). \tag{35}$$

This resulting reflectance map is normally much more directional than if the same surface were viewed by a passive imaging system at visible wavelengths.

Two different parameterizations for area A and RCS σ_o are used in the radar literature. One parameterization [31] uses the area of a pixel projected onto the surface times the normalized radar cross section. In this approach, the area factor is approximated as

$$A \approx \sqrt{1 + u_r^2 + u_y^2} = \frac{1}{\cos_i}. \tag{36}$$

The second parameterization [15] uses the area projected onto a plane normal to the illumination. The illumination area A_I denotes the area of the incident plane wave that impinges on the projected pixel. For the rectangular pixel impulse response this gives

$$A_I = \Delta r \Delta y \mid u_r \mid \tag{37}$$

providing a very simple expression in terms of the differential geometry of the surface.

Many models for the variation of σ_o as a function of incidence angle are presented in the literature [4, 6, 54] based on both empirical studies and modeling the various possible scattering mechanisms that may occur. For better or worse, radar clutter is often more directional than implied by earlier work. The generalized Lambert model $\sigma_o(\alpha_i) = \cos^k(\alpha_i)$, developed as a remedy for this, has been further generalized based on empirical data. This model [42] expresses average radar cross section as a function of angle of incidence by

$$\sigma_o(\alpha_i) = \frac{\cos^k \alpha_i}{\sin^l \alpha_i}. \tag{38}$$

It has long been suggested that (38) with k in the range of 3 to 4 may provide a reasonable model for relatively smooth surfaces such as asphalt [54], and that $l = 0$, with $k \in [1, 2]$ is appropriate for very rough surfaces. The other model [5] is derived by a physical optics approach but evaluated in the zero wavelength limit. These models define the reflectance function used [4] for extracting depth information from one or more SAR images. A typical SAR and the Lambertian reflectance maps are plotted in Figure 4.

Albedo of Radar Returns. Cosgriff [15] applied the term albedo to radiometry by analogy with photometry to mean that portion of the incident energy which is reradiated by the surface. For convenience, one can lump together all multiplicative constants that appear in front of the reflectance map and refer to that product as albedo. Albedo depends on the electrical properties of the surface materials and any ground cover. Variation in the median of measured radar albedo for typical clutter spans a wide dynamic range on the order of 60 dB where the extreme lows are for water and the extreme highs are for cities. The issue of albedo variation is so complex that only very simple models can be used.

Observation Noise. The most significant source of noise for SAR imagery is speckle. Thermal noise from the radar electronics introduces a noise source with complex amplitude modeled as white complex Gaussian noise added to the image. A third source of noise arises from the voltage accumulated in the pixel sidelobes. The combined effect is that the noisy intensity can be expressed

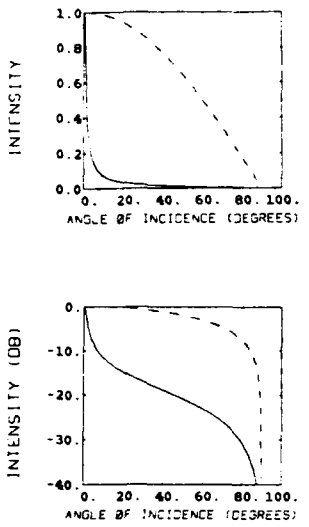


Figure 4: A typical SAR image reflectance map and Lambertian reflectance map.

as a noise-free image, combined with a bias term (dependent on thermal and sidelobe noise levels) multiplied by a noise term

$$I(r, y) = \left[R(u_r, u_y, \beta) + \sigma_{bias}^2\right] n(r, y) \tag{39}$$

where σ_{bias}^2 is the combined power or the thermal and side-lobe noise, and $n(r, y)$ is unit power random field distributed as chi-squared with two degrees of freedom. A simplified model ignoring the effects of noise for estimation of surface elevation is

$$I(r, y) = R(u_r, u_y, \beta) \tag{40}$$

3 Survey of Ongoing Work

3.1 Preprocessing

Kuan et al. [44, 45, 46] have developed smoothing algorithms for reducing speckle and signal dependent noise in gray scale images. Their image model is based on the physical process of coherent imaging and accounts for spatial correlation due to speckle, an important characteristic of speckle that has long been ignored in previous studies. Yet, their method ignores some of the complications existing in real SAR imaging systems and was only tested on real gray scale optical images corrupted by computer generated speckle noise.

Detecting edges in SAR images has also received some attention in the literature. In [72], a ratio detector is proposed. A modification to the Marr-Hildreth edge detector using a ratio of averages is discussed in [11].

3.2 Segmentation of single look complex SAR data

Owing to the modeling of SAR complex data, the segmentation problem can be formulated as an estimation problem where the region labels of individual pixels are inferred from a knowledge of the radar observations and

a priori knowledge of the distribution of the region labels. Region labels are unobserved pixel attributes that identify the region to which each pixel belongs. In [65], a Markov random field [8, 9, 13, 29, 84] has been used to model the region labels. Modeling of the observations is based on an understanding of the physics of the SAR imaging and processing system, and emphasizes the use of correlative information from neighboring pixel elements. Using Bayes' theorem the prior distribution of the radar measurements is combined with the prior distribution of the region labels to obtain the posterior distribution of the region given the radar measurements. Several possible optimality criteria [8, 14, 30, 57] that define an optimal region labelling process are then considered and implemented on an optimization network [37, 56]. Some examples of segmentation results are given in Figure 5. Figure 5(a) is the original single look intensity image of sea ice (although complex data was used for processing, intensity images are displayed) acquired in the Beaufort Sea during the Alaska campaign of March 1988 by the multifrequency polarimetric NASA-JPL airborne SAR instrument.

Figure 5(a) shows the logarithm of the magnitude of the complex-amplitude array at L-band frequency (24-cm wavelength) with a horizontal linear polarization of the transmitted and received radar signal. A direct display of the intensity array could not be used because of the large dynamic range of the pixel intensities and the strong presence of speckle. The image is 1024 x 512 pixels in size and has 3.025-m pixel spacing along azimuth [left to right in Figure 5(a)] and 6.662-m pixel spacing along slant-range [top to bottom in Figure 5(a)]. Spatial resolution, estimated from the autocorrelation coefficient of a 100 x 100 pixel area, is 14.28m along slant range and 4.95m along azimuth. The mean intensity level of four regions was selected by using an unsupervised technique [65]. Visual inspection of the SAR revealed that these four distinctive regions could be sorted into four different sea-ice types: (1) sea-ice ridges, visible as bright and fine lineaments in Fig. 5(a); (2) multilayer sea ice, a thick ice that survived several summer melts and has a relatively bright signature; (3) first-year sea ice, a newly formed ice of varying thicknesses and intermediate brightness; (4) open water and frazil ice, which appear dark in the imagery within narrow open leads. A 3×3 neighborhood window was used during segmentation to minimize boundary segmentation errors. Segmentation results using an approximate maximum a posteriori (MAP) technique [65] are given in Fig. 5(b). A selection of sample boxes across homogeneous areas of multilayer sea ice revealed segmentation accuracy of approximately 97 % for the MAP classifiers.

Derin et al. [20] developed a region-segmentation technique in which the image model consists of two levels: At the lower level, a first order causal Markov random field (MRF) describes the speckled complex amplitudes; at a higher level, a second MRF models region labelling, i.e. the grouping of pixels into spatially homogeneous regions. However, speckle is arbitrarily assumed to have a separable exponential autocorrelation function, which, as demonstrated in [65] is not appropriate for SAR complex data.



a



b

Figure 5: (a) Original single-look intensity field of SAR complex data acquired over the Beaufort Sea in Alaska. In decreasing order of brightness are ice ridges, multiyear ice, first-year ice, and open water and frazil ice. (b) MAP segmentation map.

The extension to the multifrequency case is simple. The complex amplitudes resulting from multifrequencies can be modeled as statistically independent as they correspond to independent realizations of speckle. Thus if f_1 and f_2 are two frequencies of SAR imagery with amplitudes $\mathbf{a_1}$ and $\mathbf{a_2}$ and covariance matrices $\mathbf{R_1}$ and $\mathbf{R_2}$ then the distribution of SAR complex data corresponding to the two frequencies can be written as

$$p(\mathbf{a_1}, \mathbf{a_2}) = \frac{1}{\pi^{2M}} \exp\{-\mathbf{a_1}^{*T}\mathbf{R_1}\mathbf{a_1} - \ln |\mathbf{R_1}| - \mathbf{a_2}^{*T}\mathbf{R_2}\mathbf{a_2} - \ln |\mathbf{R_2}|\} \quad (41)$$

where the $\mathbf{R_i}$ are modeled as in (12). By using simple Markov random fields for region models one can obtain posterior distributions corresponding to multifrequency data. The results of applying an approximate MAP segmentation algorithm are given in Figure 6. A single look, multifrequency log-amplitude image of terrain is shown in Figure 6(a) and 6(b). The 30 dB bandwidth of the system coherent impulse response function estimated from a 100 x 100 area is 10.63 meters in range and 4.01 meters

in azimuth at L-band HH, and 10.54 meters in range and 3.80 meters in azimuth at C-band VV. Based on the selection of thirteen training areas, the images were classified into thirteen regions. The MAP segmentation result is shown in Figure 6(c). The combined classification accuracy is about 84%. Additional results may be found in [67].

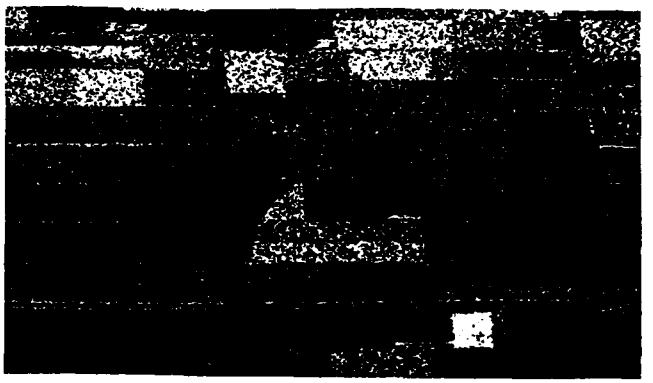
3.3 Terrain estimation

Topographic data can be extracted from SAR imagery using at least three kinds of approaches, which respectively exploit radiometric, geometric, and interferometric information. Radiometric information is useful because the backscattered power is a function of terrain surface slopes. This is the basis of shape from shading techniques. Geometric information enters through the relatively stable relationship between the scene geometry and its projection into a SAR image. This is the basis of stereogrammetry, using multiple images, and for mensuration of discrete objects in a single image. Interferometry uses the relative phase variation between the complex amplitudes of two SAR images made from two vertically separated antennas to obtain estimates of the elevation angle to the terrain surface.

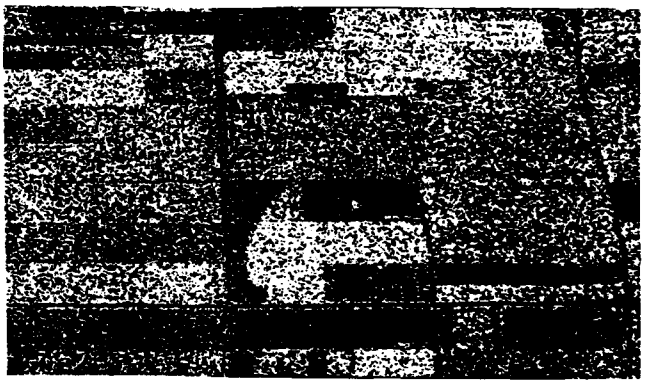
It has been suggested that all three approaches are complementary to some extent. [21]. It is well known in the computer vision community that shading of smooth surfaces and stereo parallax provide complementary information: Shading is useful in areas with constant albedo and fairly smooth variations in surface orientation. Conversely, stereo matching is more effective in areas with large albedo variations or sharp surface orientation changes that provide the required intensity features. For SAR imagery, geometric and radiometric information are not only complementary, they are synergistic. Because SAR is an active sensor, both geometric as well as radiometric disparities are created when a SAR stereo image-pair is formed. This limits the accuracy of radar-stereogrammetry [50]. The performance of interferometry depends on the ability to reliably unwrap the phase of noisy image data and correct for a variety of phase errors [52]. The point is that a single information source usually does not yield all of the topographic information available from SAR imagery.

The use of radar backscatter power to extract terrain surface orientation, sometimes referred to as radarclinometry [81], has been considered before. Cosgriff, et al. [15], while concentrating on clutter modeling for system design purposes, discussed the possibility of radarclinometry and suggested a solution method using multiple radar images. Wildey [81, 82, 83] developed algorithms for reconstructing surface topography from the shading in a single SAR image. Those algorithms directly inverted a differential equation similar to (40) for SAR imagery subject to local constraints on the relationship between z_x and z_y. These constraints are similar to, although more general than, the local sphericity assumption used by Pentland [60] and also must be inferred from image intensity derivatives.

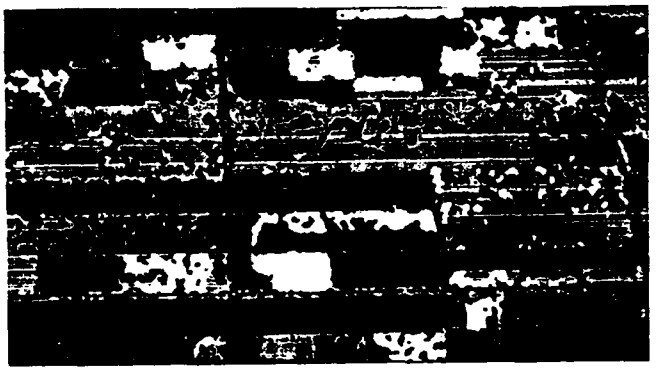
Frankot and Chellappa developed a practical method [27] for estimating the topography of natural



(a)



(b)



(c)

Figure 6: Segmentation of multifrequency SAR data. (a) Original one-look log-amplitude SAR image of Flevoland at L-band HH. (b) C-band VV. (c) MAP segmentation result.

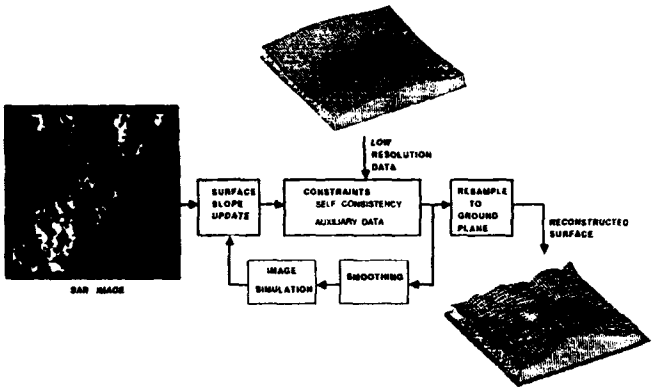


Figure 7: Block diagram of SAR SFS approach for estimation of surface elevation.

terrain from the radiometric or shading information in a SAR image. Five essential features are combined in developing the shape from shading algorithm for SAR imagery. First, an imperfect fit between the observed image intensity and the estimated surface is explicitly accounted for through a mean squared-error term. Second, a regularization penalty constraint encourages smoothness in the estimated surface. Third, self consistency of the estimated surface is enforced through an interability constraint. Fourth, the low resolution surface data has been utilized to replace low frequency surface height information not available from shading alone and to obtain estimates of the reflectance model parameters. Finally, the SAR image coordinates and reflectance models from Section 2.8 have been incorporated.

Given the SAR image and a companion digital terrain elevation model (DTM) at much lower resolution, the surface estimation procedure is as follows: The DTM is registered to the SAR imagery. Reflectance model parameters are estimated such that the mean squared-error between the SAR image and its prediction from the low resolution DTM is minimized. The function R is represented by a model with unknown parameters, dependent on both scene characteristics and sensor characteristics as discussed in Section 2.8. Given suitable estimates of those parameters, the SFS algorithm iteratively estimates the terrain surface slopes (u_r, u_y). At each iteration, integrability of those surface slopes is enforced and, by the same process, a surface height estimate is constructed.

This is accomplished using a fast least-squares procedure that involves a weighted sum of the Fourier transforms of the estimates for u_r and u_y. The Fourier transform of the DTM is substituted for the low frequency portion of the surface estimate, which are not adequately represented by shading information. This difficulty is fundamental because image intensity is a noisy function of surface height derivatives. It is exacerbated for SAR imagery because of the often highly directional nature of R and the presence of speckle noise. This approach was tested using real SIR-B SAR imagery and independently derived DTMs. The resulting surface reconstructions compared favorably with the high resolution DTM. This method has been adopted [71] for use in mapping the Venus terrain from the high resolution SAR image and the low resolution altimetry data to become available from the NASA Megallon project. A block diagram of this method is shown in Figure 7.

Zheng and Chellappa [89] later developed a method for estimation of surface topography by combining SFS and stereopsis. In this work, Horn's [38] technique is first used to reconstruct the needle maps from opposite-sided SAR stereo images. Then the needle maps are transformed to ground plane coordinates and a facet model based feature extraction algorithm [34] is used to extract topographic primal sketches. Next stereo matching is done between the feature points of two images; the disparity values and hence the depths along the feature points are computed. Subsequently, an SOR technique [10] is used to interpolate a smooth surface from the sparse depth map. Finally, the low frequency components of the scene are extracted from the reconstructed surface and a high resolution terrain map is obtained by applying the Frankot-Chellappa SFS algorithm [27].

3.4 Object Recognition

The reflective scattering from SAR causes man-made objects to be specular or mirrorlike whereas at optical wavelengths man-made objects predominantly scatter diffusely. The primary features that are observed in SAR signatures are point-like returns which predominantly arise from angled supports that serve to direct energy back towards the radar [88]. The range resolution of SAR is independent of range to the target because range resolution is a function of the waveform bandwidth. Its azimuth resolution is also independent of range.

In [88], the role of SAR sensor knowledge in indexing and pose estimation is illustrated. The three major steps considered are feature extraction, matching between prestored and predicted features, and pose estimation based on these correspondences. The feature extraction stage essentially consists of detecting local maxima that are brighter than the background. Since SAR gives point-like returns corresponding to the various scattering centers on the object, the feature extraction step captures much of the information contained in the SAR image.

A geometric hashing procedure [49] can be used for establishing a correspondence between the extracted point patterns and pre-stored patterns. When adapted to SAR, this technique is equivalent to a 2-D correlation over a sparse set of translation and rotation values using binary SAR model templates over a binary SAR image where the 1's represent the locations of the dominant scattering centers. This sparse set is the set of all translations and rotations of the template that align at least one pair of 1's in the model template and in the binary data.

After correspondence is established, the model parameters are estimated using a hierarchical extended Kalman filter [58]. At the top level in the hierarchy, the object model is represented by its state vector delineating its 3-D location and orientation. At the next level, the object components are represented as rigid body transformations referenced to the object. Associated with these

components, the locations of scattering centers and their persistence over view angle is also stored. Finally, the transformation from the 3-D world frame to the 2-D sensor/image plane is represented at the lowest level in the Kalman filter to predict where and to what accuracy the model scattering centers are located in the image. This representation of the model object is a very economical way to store the model parameter uncertainty during the estimation process.

The last step in the recognition process consists of performing a hypothesis test to determine the degree to which the data matches the constrained model. Since pose estimation has already occurred hypothesis testing occurs in the sensor frame. Extracted feature attributes are matched with predicted feature attributes using a likelihood ratio test described in [58].

A model based object recognition method that combines a bottom-up evidence accumulation process and a top-down hypothesis verification process has been presented in [47] for detection of a jet airplane in SAR images. A SAR simulator developed at The Analytic Sciences Corporation was used to generate SAR imagery. A relaxation labeling algorithm was also developed for the same task, using both point-like and line-like features [16].

In related work [68], a generic model based vision algorithm has been developed for recognizing shiny objects using their specular features utilizing a solid model, a SAR sensor model, and an image simulator for specular features. All possible patterns of specular reflections are pre-compiled and aspect maps are generated. A coarse-to-fine deformable template matching procedure is used for recognition. Results using simulated SAR images are presented.

Related significant work on target detection and identification in SAR imagery can be found in [24]. Due to lack of space we are unable to provide more details about this paper. Issues about fusion of SAR and infrared sensor data are discussed in [12].

4 Possible Research Directions

4.1 Terrain estimation

Because vision is a highly underconstrained problem, the fusion of information from different images and from different cues within a single image is important. For example, it has been suggested that SFS complements stereogrammetry [50]. By utilizing radiometric information one can improve the reliability, accuracy, and resolution of topography estimates available from radar-stereogrammetry. For SAR imagery, a tradeoff exists between two competing effects: the larger the difference in look angles for the stereo image pair the less sensitive the surface reconstruction is to errors in stereo correspondence and, on the other hand , the errors in stereo correspondence grow as the disparity between the look angles increases [50]. In many cases the human eye is not even able to recognize that two SAR images are of the same terrain if they are made from drastically different look angles because the change in illumination geometry causes changes in shading. Hence, the radiometric information contained in the images is synthetic.

It is difficult to obtain reliable stereo matches, especially in the presence of speckle noise. In order to obtain reliable matches, the resolution of the matches is necessarily low for SAR imagery. Feature-based matches require extended features and matching by correlation requires large integration subareas [25, 62] for reliability. The resolution of the matches can be improved without compromising reliability by using smaller subareas and then resolving ambiguities with shading information. The image shading predicted by the stereoscopically derived surface reconstruction should approximately fit the shading in the observed image. A secondary requirement is that the variance of the observed image intensity exceeds the variance of the stereo-predicted shading component by at least some threshold, predicted by speckle characteristics.

Similarly, the precision of stereo matches can be improved by accounting for shading differences that occur between SAR stereo image pairs. Two correction approaches, intensity prediction and intensity compensation, are possible. The intensity prediction method extracts high frequency surface information using SFS techniques and then forms a predicted image. Stereo matching of, say, the first SAR image with its prediction from a second SAR image allows the computation of a residual parallax error, used for estimating a residual surface. The intensity compensation method starts with a stereoscopically derived surface reconstruction and then predicts a local shading ratio between the two images. Given the surface reconstruction, the aspect difference between the two images, knowledge of the parallax errors, and the reflectance map it is possible to compute the shading compensation. If additive noise terms are low, the correction is insensitive to albedo variations. After shading compensation the stereo matching procedure is repeated with greater achievable accuracy.

One can also experiment with methods that combine SFS stereo methods by explicitly including the surface height [39, 90] $u_1(r, y)$ in the SFS equations. Another method [69] may be to use generate needle maps for each image using an appropriate SFS technique. The depth map can then be generated from these needle maps by establishing the correspondence so that the disparity over these needle maps is minimized. The integrability constraint can also be enforced.

Radiometric stereo can also be extended to SAR imagery and applied simultaneously with traditional geometric stereo. In applying radiometric stereo and in the above intensity prediction and correction methods to SAR imagery the problem of surface-height dependent registration errors arises. The local registrations provided by the initial stereo correspondences may be sufficiently accurate to allow radiometric stereo, geometric stereo and monocular shading cues to bootstrap each other. Recent methods for extracting stereo depth maps from visual images [3] can be re-evaluated for SAR imagery, to provide better methods for utilizing both radiometric and geometric information.

Earlier work has considered the estimation of surface height when the reflectance properties of the surface are

constant. The estimation of surface topography simultaneously with variations in albedo and other surface properties, such as roughness, is a much more difficult problem. Multi-spectral Landsat imagery has been used to segment albedo variations independent of surface topography [23, 35]. In a similar manner, one can detect variations in albedo using multifrequency radar imagery [7, 77]. The recent analysis of multi-polarization radar imagery [77] may provide insight into the utilization of phase data and polarization diversity for inferring surface structure. For example, it is possible to distinguish between surface and volume scattering mechanisms, and therefore reflectance maps class, using the phase differences between images sensed with different polarizations.

More research is needed in order to make use of these additional information sources. Models are needed which are general enough to account for the effect of surface roughness, the dielectric properties of the surface, polarization diversity, and frequency diversity, yet are tractable enough to be useful for image analysis.

4.2 Segmentation of Multilook Intensity and Multipolarimetric SAR Data

As mentioned in the introduction, it is desirable to be able to adapt SAR algorithms to multilook intensity imagery. Kelly et al. [41] have developed an adaptive technique for segmentation of speckled images using a hierarchical random field model. However, the joint probability distribution function of a set of M N-look intensities does not have a simple analytical form. Even if it were possible to derive a closed form expression of the joint density based on a marginal gamma distribution of the intensity, the result would not be relevant. The samples that are used to generate the N-look image are correlated; therefore the equivalent number of looks in terms of reducing the variance of the signal is not equal to N but slightly inferior to N. Furthermore, the marginal distribution of a N-look intensity is not **gamma** distributed. By feeding in the value of the equivalent number of looks into the analytical expression of a gamma distribution, one gets only an approximation to the correct distribution, as noted in [33]. To conclude, the computation of the joint probability density function of M N-look intensities appears to be hopelessly complicated if an exact solution is desired, and quite unreliable if a gamma distribution assumption is used for the marginal distributions.

The only valid results concern the derivation of the variance and of the autocorrelation function of the intensity field. The variance of speckle in multilook imagery is equal to the variance of speckle in one-look imagery divided by the equivalent number of looks. The correlation matrix $\mathbf{R_{I0}}$ can be modeled based on the correlation properties of the complex amplitudes used to generate the multilook intensities. If we assume that the correlation properties of each of the N intensities are identical (approximation), and there is no overlap between the look-filters, the resulting correlation matrix of the sum of these intensities will be sensibly the same as the correlation matrix of one intensity computed from one sub-aperture. A complete and in-depth analysis of the correlation properties of multilook intensities is also available in [55].

In terms of the joint probability distribution function itself, the only applicable result is that the statistics tend to normality for large N. Although it is known that in practice this is not a solid assumption until N is of the order of several tens, one can make the approximation of normality starting at N = 4. The joint probability distribution function of a set of M zero-mean intensities is therefore

$$p(I_1^0,\ldots,I_M^0) = \frac{1}{(2\pi)^{M/2}|\mathbf{R_{I0}}|^{1/2}} \exp\{-\frac{1}{2}\mathbf{I}^{0*T}\cdot\mathbf{R}_{\mathbf{I0}}^{-1}\cdot\mathbf{I}^0\} \tag{42}$$

where

$$\begin{aligned} \mathbf{I}^0 &= [I_1^0,\ldots,I_M^0] \\ \mathbf{R_{I0}} &= < I_m^0 \cdot I_n^0 >= |R_a|^2 \\ I_i^0 &= I_i - < I_i >, \qquad \forall i \in \{1,\ldots,M\} \end{aligned}$$

and R_a is given in (17).

By using the model [65] for the region labelling process, MAP, ICM and MPM estimates of the labelling process can be computed. The methods are similar to the complex data case and involves 1/8 as many operations for the same area coverage.

From the conditional distribution of a single polarimetric measurement vector (28), one can derive the joint conditional distribution of a small set of contiguous polarimetric measurement vectors contained in a window N_s and centered at site s. Because of the presence of the SAR system impulse response function, part of this additional information is already contained in $\mathbf{X}_s$, but for computational convenience one can assume that the M polarimetric measurement vectors $\mathbf{X}_s = [X_1,\ldots,X_M]$ corresponding to the pixel elements of N_s are spatially uncorrelated. As the polarimetric complex amplitudes are circular Gaussian, they are independent and the conditional distribution of X_s is the product of the marginal distributions. Again, one can view the polarimetric data array as composed of a set of overlapping windows of M elements centered at each pixel site s such that each of these windows is homogeneous (all pixels have the same region label), and conditionally independent of the other windows. The joint conditional distribution of $\mathbf{X}_s$ is then expressed, using a Gibbs representation, as

$$p(\mathbf{X}_s) = \frac{\exp\{-MU_1^s(\mathbf{X}_s)\}}{\pi^{3M}} \tag{43}$$

where the energy function U_1^s is

$$\begin{aligned} U_1^s(\mathbf{X}_s) &= \frac{|HH^M|^2}{\sigma(1-|\rho|^2)} + \frac{|HV^M|^2}{\sigma\epsilon} + \frac{|VV^M|^2}{\sigma\gamma(1-|\rho|^2)} \\ &- 2\epsilon\sqrt{\gamma}\,|HH^M||VV^M||\rho| \end{aligned}$$

$$\cos\phi(\phi_{HHVV^M} - \phi_{\rho l} + \log\sigma^3\epsilon\gamma(1-|\rho|^2) \tag{44}$$

with

$$HH^M VV^{M*} = \tfrac{1}{M}\textstyle\sum_{i=1}^{i=M} HH_i VV_i^*$$

$$| HH^M |^2 = \frac{1}{M} \sum_{i=1}^{i=M} | HH_i |^2 \quad (45)$$

The operation of computing (45) is equivalent to a M-look operation performed on polarimetric SAR data [22]. As (44) does not depend on the absolute phase of each component of the polarimetric measurement, the same algorithm can therefore be used for both single look and multilook SAR polarimetric complex data.

The segmentation technique of [43] suffers from the shortcoming of only using a multispectral description. One can modify the technique so that region level descriptions can be added to improve segmentation.

The major difficulty is to be able to simply model the terms of the covariance matrix. The estimation of the polarimetric parameters e, ρ, and γ has also been acknowledged as being difficult. A simple model used in conjunction with a multilevel Markov random field is adequate to represent the labelling process and the same technique used for SAR complex data can be used to compute the MAP or the MPM estimate of the labelling process. One can also study the effect of reducing the dimension of the polarimetric measurement vector on segmentation accuracy.

Analyzing multifrequency polarimetric data represents a real challenge to the experimentalist because of the high dimensionality of the data. It is also a challenge for scientists because little has been done in terms of classifying multifrequency polarimetric data. Most of the early analysis has been through visual interpretation. If the segmentation of polarimetric SAR data can be performed by the aforementioned technique, it would then be simple to add other frequencies to the data to improve feature detection (since multifrequency data are independent) so that the algorithm can be used to segment multifrequency polarimetric SAR data.

4.3 Change detection in SAR images

4.3.1 Difference versus ratio

Given a pair of SAR images of the same scene acquired at two different dates, a conjecture is to determine whether it is better to use the ratio or the difference of the SAR intensities to measure the degree of change in the backscattering properties of the target.

In single look SAR intensity data, the distribution of the intensity is

$$p(I_1/ < I_1 >) = \frac{1}{\pi} \exp -\{\frac{I_1}{< I_1 >} + \log < I_1 >\} \quad (46)$$

at a time t_1 for a given homogeneous region. A similar expression can be written at a different observation time t_2.

If there is no change in the physical and electrical characteristics of the surface between the two dates and if the imaging conditions are identical, image speckle will be the same in both images. If the data are coregistered with sub-pixel accuracy, speckle will be exactly repeated pixel by pixel as the distribution of scatters in each resolution cell will be the same. This is in fact the fundamental principle of SAR interferometry which has been verified as valid even in the case of SAR data acquired several days apart [32]. In those circumstances, the different between the two intensities is independent of image speckle.

By contrast, if a change occurs in the physical and electrical characteristics of the surface, i.e. the distribution of scatterers and the resulting backscatter response are different, image speckle will be uncorrelated between the two dates. As image speckle has circular Gaussian statistics, we may thereby assume that, for change detection, multitemporal realizations of speckle are mathematically independent. The joint distribution of the two observations is therefore

$$\begin{aligned} & p\,(I_1, I_2/ < I_1 >, < I_2 >) \\ & = p\,(I_1/ < I_1 >)\, p\,(I_2/ < I_2 >) \end{aligned} \quad (47)$$

The distribution function of the difference $d = I_2 - I_1$ is derived by integrating the joint distribution over I_1 leading to

$$\begin{aligned} & p(d/ < I_1 >, < I_2 >) \\ & = \frac{1}{<I_1>+<I_2>} \exp\{-d/ < I_2 >\} \end{aligned} \quad (48)$$

which is similar to the distribution of the original speckled intensities. Similarly, letting $r = I_2/I_1$, the distribution of the ratio is derived after a change in variables by integrating the joint distribution over I_1 leading to

$$\begin{aligned} & p(r/ < I_1 >, < I_2 >) \\ & = \frac{<I_2>/<I_1>}{(r + <I_2>/<I_1>)^2} \end{aligned} \quad (49)$$

If we assume a threshold d_0 to decide on whether to classify a given difference d between class A characterized by a difference $d_A =< I_A > - < I_1 >$ or class B characterized by a difference $d_B =< I_B > - < I_1 >$ (i.e. $d > d_A$ is classified as class B, and $d \leq d_0$ is classified as class A), the class A probability of error is given by

$$\begin{aligned} PE_A &= \int_{d_0}^{\infty} p(d/d_A) dd \\ &= \frac{< I_A > / < I_1 >}{1 + < I_A > / < I_1 >} \exp\{-d_0/ < I_A >\} \end{aligned}$$

and the class B probability of error by

$$\begin{aligned} PE_B &= \int_0^{d_0} p(d/d_B) dd \\ &= \frac{< I_B > / < I_1 >}{1 + < I_B > / < I_1 >} \\ &\quad (1 - \exp\{-d_0/ < I_B >\}) \end{aligned} \quad (50)$$

Assuming equal a prior probabilities for class A and class B, the threshold decision is found using

$$p(d_0/d_A) = p(d_0/d_B) \quad (51)$$

leading to

$$d_0 =< I_A >< I_B > \frac{\log[\frac{1+<I_B>/<I_1>}{1+<I_A>/<I_1>}]}{< I_B > - < I_A >} \quad (52)$$

The total probability of error is then computed as

$$PE = \frac{1}{2}(PE_A + PE_B) \quad (53)$$

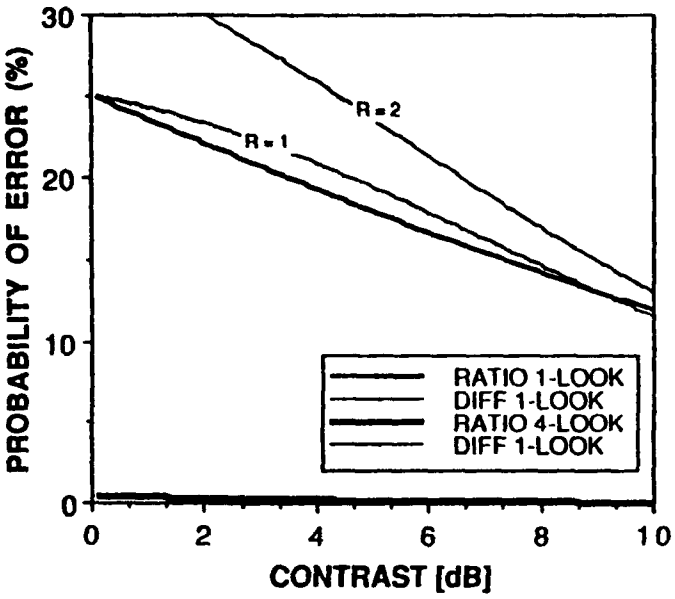


Figure 8: Probability of error of the difference and of the ratio for single-look and four-look SAR data.

In the case of the ratio, with $r_A = < I_A > / < I_1 >$ and $r_B = < I_B > / < I_1 >$, the decision threshold is

$$r_0 = \frac{\sqrt{< I_A >< I_B >}}{< I_1 >} \tag{54}$$

and the probability of error is

$$PE = \frac{\frac{1}{2}}{1 + \sqrt{< I_A > / < I_B >}} \tag{55}$$

The probability of error using the ratio only depends on the ratio of the two possible changes (i.e. $< I_A > / < I_B >$), whereas the probability of error using the difference also depends on the absolute change of each class (i.e. $< I_A > / < I_1 >$ or $< I_B > / < I_1 >$). A plot of two probabilities of error is shown in Figure 8 as a function of the contrast between the possible classes $< I_A > / < I_B >$, for $< I_1 >= 1$, and where $R = < I_B > / < I_1 >$. The results illustrate the better performance of the ratio independent of the intensity level I_1 of the region.

The same computation can be done using a gamma distribution of the intensity, leading to a distribution of the ratio

$$p(r) = \frac{\Gamma(2N-1)}{(\Gamma(N))^2 N^{2N}} \frac{(< I_2 > / < I_1 >)^N r^{N-1}}{(r + < I_2 > / < I_1 >)^{2N}} \tag{56}$$

assuming that the number of looks N remains the same. The decision threshold is equal to the decision threshold of the single look case. For $N = 4$, the corresponding probability of error is shown on Figure 8 and illustrates the significant improvement in detection accuracy obtained when multilook SAR intensity data are used rather than single look SAR data.

A Bayes classifier for change detection

Multitemporal SAR data can be segmented into regions of homogeneous and similar change in radar backscatter. As before, we view the image array Ω as composed of a set of conditionally independent overlapping windows N_s of M elements centered at each pixel s such that all the pixels contained in the window have the same region label. The joint distribution function of the M backscatter ratios $\mathbf{r}_s = [r_1, \ldots, r_M]$ contained in a neighborhood N_s of site s is approximated by the product of the marginal distributions, i.e.

$$p(\mathbf{r}_s / L_s) = \prod_{i=1}^{i=M} p(r_i / L_s) \tag{57}$$

The corresponding likelihood function is

$$\ln[p(\mathbf{r}_s / L_s)] \propto M\, U_1(\mathbf{r}_s / L_s) \tag{58}$$

where

$$U_1(\mathbf{r}_s / L_s) = \frac{1}{M} \sum_{i=1}^{i=M} [(N-1) \ln r_i - 2N \ln(r_i + r_l)] + N \ln r_l \tag{59}$$

r_l is the ratio of region $L_s = l$, and N is the number of looks.

The distribution of the region labels is given as

$$p(L_s / L_r, r \in N_s^o) \propto \exp -\{M\, U_2(L_s / L_r, r \in N_s^o)\} \tag{60}$$

The logarithm of the posterior distribution of the region labels is

$$\ln[p(L_s / \mathbf{r}_s, L_r, r \in N_s^o)] \propto U_1(\mathbf{r}_s / L_s) + U_2(L_s / L_r, r \in N_s^o) \tag{61}$$

Using the approximate MAP criterion, optimal region labelling of the image is defined as minimizing an energy function

$$E_{\mathrm{MAP}} = \sum_s (U_1(\mathbf{r}_s / L_s) + U_2(L_s / L_r, r \in N_s^o)) \tag{62}$$

The optimization techniques discussed in [65] again are applicable to compute an approximate MAP or other solutions.

The feature vector used for clustering will be the difference between the logarithm of the components of the covariance matrix of each data sample. Clustering techniques [1] can be used to cluster the sample measurements.

4.3.2 Multitemporal segmentation of SAR data

Change detection only addresses the relative change in the backscatter characteristics of the surface between two different dates. In some circumstances, it may be useful to consider the absolute values of the backscatter returns and classify the data into regions of homogeneous change and homogeneous backscatter characteristics. Such a strategy may prove useful for difficult segmentation problems where single observations cannot separate different types of natural surfaces, e.g. different types of terrain.

In that case, the joint distribution of the multitemporal radar measurements must be used instead of the distribution of the backscatter power ratio. The technique is similar to the one adopted for the segmentation of multifrequency SAR data.

5 Conclusion

We have discussed a number of important and interesting problems related to analysis and interpretation of SAR imagery. Much more research needs to be done for effective transfer of IU methodology to the SAR domain.

Acknowledgement

The first author wishes to acknowledge the contributions of Dr. R.T. Frankot to Sections 2.8, 3.3 and 4.1.

References

[1] Ball G.H. and Hall D.J., "A Clustering Technique for Summarizing Multi-Variate Data", *Beh. Sci.*, vol. 12, pp. 153–155, 1967.

[2] Barber B.C., "Theory of Digital Imaging from Orbital Synthetic Aperture Radar", *Int. J. Rem. Sens.*, vol. 6, pp. 1009–1057, July 1985.

[3] Barnard S.T., "A Stochastic Approach to Stereo Vision", AIAA Fifth Conf. on Artificial Intelligence, Philadelphia, PA, pp. 676–680, August 1986.

[4] Barrick D.E., "Rough Surface Scattering Based on the Specular Point Theory", *IEEE Trans. Antennas Propagat.*, vol. AP-16, pp. 838–850, July 1968.

[5] Barrick D.E., "Relationship Between Slope Probability Density Function and the Physical Optics Integral in Rough Surface Scattering", *Proc. IEEE*, vol. 56, pp. 1728–1729, Oct. 1968.

[6] Barton D.K., Ed., *Radars—Volume V, Radar Clutter*, Artech House, MA, 1975.

[7] Benton A.R. and Newton R.W., "The Utility of Multifrequency Radar in the Coastal Zone Environment", *IEEE Geosci. Rem. Sensing Soc. Newsletter*, vol. 9, pp. 4–8, June 1985.

[8] Besag J.E., "On the Statistical Analysis of Dirty Pictures", *J. Roy. Statist. Soc.*, Ser. B, vol. 48, pp. 259–302, 1986.

[9] Besag J.E., "Spatial Interaction and the Statistical Analysis of Lattice Systems", *J. Roy. Statist. Soc.*, Ser. B, vol. 36, pp. 192–236, 1974.

[10] Blake A. and Zisserman A., *Visual Reconstruction*, MIT Press, Cambridge, MA, 1987.

[11] Bovik A.C., "On Detecting Edges in Speckle Imagery", *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-36, pp. 1618–1627, Oct. 1988.

[12] Bullock M. et al., "SAR and IR Data Fusion Using the Sensor Algorithm Research Expert System (SARES)", SPIE Intl. Conf. on Aerospace Sensing, Orlando, FL, April 1991.

[13] Chellappa R., "Two-dimensional Discrete Gaussian Markov Random Field Models for Image Processing", in L.N. Kanal and A. Rosenfeld, Eds., *Progress in Pattern Recognition*, 2, pp. 79–112, North-Holland, Amsterdam, 1985.

[14] Cohen F.S. and Cooper D.B., "Simple Parallel Hierarchical and Relaxation Algorithms for Segmenting Noncausal Markovian Random Fields", *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-9, pp. 195–219, March 1987.

[15] Cosgriff R.L., Peake W.L., and Taylor R.C., "Terrain Scattering Properties for Sensor System Design", Tech. Report, Engineering Experiment Station, Ohio State University, Columbus, 1960.

[16] Cucka P. and Rosenfeld A., "Linear Feature Compatibility for Pattern-Matching Relaxation", Center for Automation Research Technical Report CS-TR-2579, University of Maryland, Jan. 1991.

[17] Curlander J.C. and McDonough R., "Synthetic Aperture Radars—Systems and Signal Processing", in press, 1991.

[18] Curlander J.C., Kwok R. and Pang S., "A Post-Processing for Automated Rectification and Registration of Spaceborne SAR Imagery", *Int. J. Rem. Sens.*, vol. 8, pp. 621–638, 1987.

[19] Curlander J.C., "Location of Spaceborne SAR Imagery", *IEEE Trans. Geosci. Rem. Sens.*, vol. GES-20, pp. 359–364, July 1982.

[20] Derin H., Kelly P.A., Vezina G. and Labitt S.G., "Modeling and Segmentation of Speckled Images Using Complex Data", *IEEE Trans. Geosci. Rem. Sens.*, vol. GES-28, pp. 76–87, 1990.

[21] Domik G., "Methods and Application of Surface Shape Reconstruction from Multiple SAR Images", Intl. Geosci. Rem. Sens. Symp., Ann Arbor, MI, pp. 1551–1555, May 1987.

[22] Dubois P., et al., "Data Volume Reduction for Imaging Radar Polarimeter", Proc. of IEEE AP-S Int. Symposium, vol. III, San Jose, CA, pp. 1354–1357, June 1989.

[23] Eliason, P.T. et al., "Extraction of Topographic and Spectral Albedo Information from Multispectral Images", *Photogram. Engin. Rem. Sens.*, vol. 47, pp. 1571–1579, 1981.

[24] Ettinger G. et al., "Bayesian Inference for Model-Based Vision" (preprint).

[25] Frankot R.J., "SAR Image Registration by Multi-Resolution Correlation", Proc. SPIE, vol. 423, pp. 195–203, August 1983.

[26] Frankot R.T. and Chellappa R., "A Method for Enforcing Integrability in Shape from Shading Problem", *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-10, pp. 439–451, July 1988.

[27] Frankot R.T. and Chellappa R., "Estimation of Surface Topography in Synthetic Aperture Radar Images Using Shape from Shading Techniques", *Artificial Intelligence*, vol. 43, pp. 271–310, June 1990.

[28] Freeman T., Yuhsyen S. and Werner C., "Polarimetric SAR Calibration Experiment Using Active Radar Calibrators", *IEEE Trans. Geosci. Rem. Sens.*, vol. GES-28, pp. 224–240, March 1990.

[29] Geman S. and Graffigne C., "Markov Random Field Image Models and Their Applications to Computer Vision", Proc. Int. Congress of Mathematicians, Berkeley, CA, pp. 1496–1517, 1986.

[30] Geman S. and Geman D., "Stochastic Relaxation, Gibbs Distributions, and Bayesian Restoration of Images", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-6, pp. 721–741, Nov. 1984.

[31] Goldstein H., "Frequency Dependence of the Properties of Sea Echo", *Phys. Rev.*, vol. 70, pp. 938–986, Dec. 1946.

[32] Goldstein R.M., Zebker H.A. and Werner C.L., "Satellite Radar Interferometry: Two-dimensional Phase Unwrapping", *Radio Science*, vol. 23, pp. 713–720, 1988.

[33] Goodman J.W., "Statistical Properties of Laser Speckle Patterns", in J.C. Dainty, Ed., *Laser Speckle and Related Phenomena*, Springer Verlag, Heidelberg, 1975.

[34] Haralick R.M., Watson L.T., and Laffey T.J., "The Topographic Primal-Sketch", *Intl. J. Robotics Research*, vol. 2, pp. 50–72, Spring 1983.

[35] Haralick R.M., Campbell J.B., and Wang S., "Automatic Inference of Elevation and Drainage Models from a Satellite Image", *Proc. IEEE*, vol. 73, pp. 1040–1053, June 1985.

[36] Heckbert P.S., "Survey of Texture Mapping", *IEEE Computer Graphics and Applications*, vol. 6, pp. 56–67, Nov. 1986.

[37] Hopfield J.J. and Tank D.W., "Neural Computation of Decisions in Optimization Problems", *Biol. Cybern.*, vol. 52, pp. 114–152, 1985.

[38] Horn B.K.P. and Brooks M.J., "The Variational Approach to Shape from Shading", *Computer Vision, Graphics, Image Processing*, vol. 33, pp. 174–208, Feb. 1986.

[39] Horn B.K.P., "Height and Gradient from Shading", Memo 1105, MIT AI Laboratory, May 1989.

[40] Jao J.K., "Amplitude Distribution of Composite Terrain Radar Clutter and the K-distribution", *IEEE Trans. Antennas Propagat.*, vol. AP-32, pp. 1049–1062, Oct. 1984.

[41] Kelly P.A., et al., "Adaptive Segmentation of Speckle Images Using a Hierarchical Random Field Model", *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-36, pp. 1628–1641, Oct. 1988.

[42] Keydel W., "Application and Experimental Verification of an Empirical Backscattering Cross Section Model for the Earth's Surface", *IEEE Trans. Geosci. Rem. Sens.*, vol. GES-20, pp. 67–71, Jan. 1982.

[43] Kong J.A., et al., "Identification of Terrain Cover Using the Optimum Polarimetric Classifier", *J. Electromag. Wave Appl.*, vol. 2, pp. 171–194, Feb. 1988.

[44] Kuan D.T., Sawchuk, A.A., Strand T.C. and Chavel P., "MAP Speckle Reduction Filter for Complex Amplitude Speckle Images", Proc. IEEE Conf. Patt. Anal. Machine Intell., Las Vegas, NV, pp. 58–63, June 1982.

[45] Kuan D.T., Sawchuk, A.A., Strand T.C. and Chavel, P., "Adaptive Noise Smoothing Filter for Images with Signal-Dependent Noise", *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-2, pp. 165–167, March 1985.

[46] Kuan D.T., Sawchuk A.A., Strand T.C. and Chavel P., "Adaptive Restoration of Images with Speckle", *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-35, pp. 373–383, March 1987.

[47] Kuno Y., Ikeuchi K. and Kanade T., "Model-Based vision by Cooperative Processing of Evidences and Hypotheses Using Configuration Spaces", SPIE Proc. on Digital and Optical Shape Representation for Pattern Recognition, Miami, FL, May 1988.

[48] Kwok R., et al., *Multisensor Image Registration: A Progress Report*, Jet Propulsion Laboratory Internal Report JPL D-6697, Sept. 1989.

[49] Lamdan Y. and Wolfson H., "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme", Proc. Second Intl. Conf. on Computer Vision, Tarpon Springs, FL, Dec. 1988.

[50] Leberl F. et al., "Multiple Incident SIR-B Experiment Over Argentina: Stereo-Radargrammetric Analysis", *IEEE Trans. Geosci. Rem. Sens.*, vol. GES-24, pp. 482–492, July 1986.

[51] Lewis A.J., and Mac Donald H.C., "Interpretive and Mosaicking Problems of SLAR Imagery", *Rem. Sens. Environ.*, vol. 1, pp. 213–218, 1970.

[52] Li F. and Goldstein R., "Studies of Multi-Baseline Interferometric Synthetic Aperture Radars", Intl. Geosci. Rem. Sens. Symp., Ann Arbor, MI, pp. 1545–1550, May 1987.

[53] Lim H.H., et al., "Classification of Earth Terrain Using Polarimetric Synthetic Aperture Radar Images", *J. Geophys. Res.*, vol. 94, pp. 7049–7057, June 1989.

[54] Long M.W., *Radar Reflectivity of Land and Sea*, Artech House, MA, 1983.

[55] Madsen S.M., "Speckle Theory: Modeling, Analysis, and Applications Related to Synthetic Aperture Radar Data", Ph.D. thesis LD 62, Electromagnetics Institute, Technical University of Denmark, Lyngby, 1986.

[56] Manjunath B.S., Simchony T. and Chellappa R., "Stochastic and Deterministic Networks for Texture Segmentation", *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-38, pp. 1039–1049, June 1990.

[57] Marroquin J., Mitter S., and Poggio T., "Probabilistic Solution of Ill-posed Problems in Computer Vision", Proc. Image Understanding Workshop, Miami Beach, FL, pp. 293–309, 1985.

[58] Morgan D., Miltonberger T. and Orr G., "A Sensor Algorithm Expert System", Proc. SPIE/SPSE Symposium on Electronic Imaging, Los Angeles, CA, Jan. 1989.

[59] Oliver C.J., "The Interpretation and Simulation of Clutter Textures in Coherent Images", *Inverse Problems*, No. 2, pp. 481–518, 1986.

[60] Pentland A.P., "Local Shading Analysis", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-16, pp. 170–187, March 1986.

[61] Porcello L.J., et al., "Speckle Reduction in Synthetic Aperture Radars", *J. Opt. Soc. Am.*, vol. 66, pp. 1305–1311, 1976.

[62] Ramapriyan, H.K. et al., "Automated Matching of Pairs of SIR-B Images for Elevation Mapping", *IEEE Trans. Geosci. Rem. Sens.*, vol. GES-24, pp. 462–472, July 1986.

[63] Radar Vision Workshop held at Santa Cruz, CA, August 1987: Speaker Presentation Slides, Compiled by D.L. Milgram.

[64] Raney R.K., "Transfer Functions for Partially Coherent SAR Systems", *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-19, pp. 740–750, Sept. 1983.

[65] Rignot E. and Chellappa R., "Segmentation of Complex Synthetic Aperture Radar Data", *J. Opt. Soc. Am. A*, vol. 8, pp. 1499–1509, Sept. 1991.

[66] Rignot E., et al., "Multisensor Registration: Requirements and Techniques", *Photogram. Engin. Rem. Sens.*, vol. 56, Aug. 1991.

[67] Rignot E. and Chellappa R., "Segmentation of Multifrequency Synthetic Aperture Radar Complex Data", Proc. Int. Conf. on Acoust., Speech, Signal Proc., Toronto, Canada, May 1991.

[68] Sato K., Ikeuchi K. and Kanade T., "Determining Object Configuration Using Specular Features", Proc. 10th Intl. Conf. on Patt. Rec., Atlantic City, NJ, June 1990.

[69] Shao M., Chellappa R., and Simchony, T., "Reconstructing a Depth Map from One or More Images", *Computer Vision, Graphics, Image Processing: Image Understanding*, vol. 53, pp. 219–226, March 1991.

[70] Skolnik M.I., Ed. *Radar Handbook*, McGraw-Hill, New York, 1970.

[71] Thomas J., Kober W. and Leberl F., "Multiple-Image SAR Shape from Shading", Intl. Geosci. Rem. Sens. Symp., 1989.

[72] Touzi R. et al.., "A Statistical and Geometrical Edge Detector for SAR Images", *IEEE Trans. Geosci. Rem. Sens.*, vol. GES-26, pp. 764–773, Nov. 1988.

[73] Ulaby F.T. and Elachi C., *Radar Polarimetry for Geoscience Applications*, Artech House, MA, 1990.

[74] Ulaby F.T., Moore R.K. and Fung A.K., *Microwave Remote Sensing*, vol. I, Addison Wesley, Reading, MA, 1981

[75] Ulaby F.T., Moore R.K. and Fung A.K., *Microwave Remote Sensing*, vol. II, Addison Wesley, Reading, MA, 1982.

[76] Ulaby F.T., Moore R.K. and Fung A.K., *Microwave Remote Sensing*, vol. III, Artech House, MA, 1985.

[77] Ulaby F. et al., "Relating Polarization Phase Difference of SAR Signals to Scene Properties", *IEEE Trans. Geosci. Rem. Sens.*, vol. GES-25, pp. 83–92, Jan. 1987.

[78] van Zyl J.J., "Calibration of Polarimetric Radar Images Using Only Image Parameters and Trihedral Corner Reflector Responses", *IEEE Trans. Geosci. Rem. Sens.*, Vol GES-28, pp. 224–240, May 1990.

[79] van Zyl J.J., "Unsupervised Classification of Scattering Behavior Using Radar Polarimetry Data", *IEEE Trans. Geosci. Rem. Sens.*, vol. GES-27, pp. 36–44, Jan. 1989.

[80] van Zyl J.J., Zebker H.A. and Elachi C., "Imaging Radar Polarization Signature: Theory and Observation", *Radio Sci.*, vol. 22, pp. 529–543, 1987.

[81] Wildey R.L., "Topography from Single Radar Images", *Science*, vol. 224, pp. 153–156, April 1984.

[82] Wildey R.L., "Radarclinometry for the Venus Radar Mapper", *Photogram. Engin. Rem. Sens.*, vol. 52, pp. 41–50, Jan. 1986.

[83] Wildey R.L., "Radarclinometry", *Earth, Moon, and Planets*, vol. 36, pp. 217–247, 1986.

[84] Woods J.W., "Two-Dimensional Discrete Markovian Random Fields", *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 232–240, March 1972.

[85] Wu C., Liu K.Y., and Jin M., "Modeling and a Correlation Algorithm for Space-borne SAR Signals", *IEEE Trans. Aerosp. Elect. Syst.*, vol. AES-18, pp. 563–575, 1982.

[86] Yueh H.A., et al., "Bayes Classification of Terrain Cover Using Normalized Polarimetric Data", *J. Geophys. Res.*, vol. 93, pp. 15261–15267, Dec. 1988.

[87] Zebker H.A., van Zyl J.J., and Held D.N., "Imaging Radar Polarimetry from Wave Synthesis", *J. Geophys. Res.*, vol. 92, pp. 683–701, 1987.

[88] Zelnio E.G., "The Importance of Sensor Models to Model-Baseed Vision Applications", in H.N. Nasr, Ed., *Automatic Object Recognition*, SPIE Optical Engineering, 1991.

[89] Zheng Q. and Chellappa R., "Estimation of Surface Topography from Stereo SAR Images", Proc. Intl. Conf. Acoust., Speech, Signal Proc., Glasgow, Scotland, May 1989.

[90] Zheng Q. and Chellappa R., "Estimation of Illuminant Direction, Albedo and Shape from Shading", *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-13, pp. 680–702, July 1991.

Image Understanding Research For Automatic Target Recognition

Bir Bhanu
College of Engineering
University of California
Riverside, CA 92521

Terry L. Jones
Center for Night Vision and Electro-Optics
AMSEL-RD-NV-ISPD
Fort Belvoir, VA 22060

Abstract

Automatic Target Recognition (ATR) is an extremely important capability for DoD applications. In this paper we discuss the problems in developing real-world ATR systems and present the status of technology for these systems. We identify some image understanding problems that need to be solved in order to enhance the effectiveness of ATR-based weapon systems. The technological gains will also lead to significant advances in other areas of applications of image understanding.

1 Introduction

Automatic Target Recognition (ATR) is the process of automatic target acquisition and classification. All the services need ATR capability [8] for a variety of missions and scenarios such as air-to-ground, ground-to-ground, surface-to-surface, air-to-air, etc. ATR systems that can work in multiscenarios do not yet exist. The development of automatic target recognizers that can perform effectively in dynamic environmental conditions will be of great importance [1].

The generic ATR problem is to take information from one or more sensors, and if necessary, combine it with *a priori* information (e.g., tactics, digital map information, etc.). A decision is then made about which targets are present. The targets are possibly prioritized by their tactical importance so that action can be taken to eliminate them. The sensors can be any one of a variety of types – acoustic, seismic, visible or near infrared, laser, millimeter wave, thermal imagers, radar, etc. – or some combination of them. The weapon platforms can be aircraft, missile, ship, or ground vehicles. The targets can be fixed (e.g., storage depots, bridges, ships in ports, etc.) or mobile (e.g., tanks, helicopters, ships at sea, etc.).

The term ATR includes both autonomous and aided recognition (or cueing with a "person in the loop"). In cueing, the acquisition is done by the targeting system, but ultimately recognition is done by the person. Although many researchers would like to perform a wide variety of missions autonomously, the services will only automate critical operator functions reluctantly. There is a built-in bias toward the flexibility of the human operator (e.g, the Air Force still relies on manned, strategic nuclear bombers despite excellent land and sea based strategic missiles). There is more willingness to remove the operator in the missions where survivability of a human is low. Soldiers may be moved further from the "action," but we do not expect them to relinquish control. Aided systems with a "person in the loop" will be employed before autonomous ones.

It is clear that ATR is a multidisciplinary area that requires diverse technology and expertise in sensors, processing, architecture, implementation, and evaluation of software and hardware systems. The related computer vision and pattern recognition technology and systems have evolved from using the statistical pattern recognition approaches to model-based vision, to knowledge-based systems. Recently, adaptive and learning systems focused on parts of ATR problems are also being developed in laboratories.

In this paper, we first present typical ATR applications for various services in Section 2. This is followed by a discussion of technological problems in developing real-world ATR systems in Section 3. Section 4 presents the current status of technology for ATR systems. In Section 5 we identify and discuss technical research areas related to ATR and image understanding. Finally, in section 6 we present the conclusions of this paper.

2 ATR Applications

Several Army applications needing ATR technology are fire-and-forget anti-tank missiles for the infantry, smart minefields, targeting for artillery, air defense, tank commander and gunner assistance, perimeter surveillance, and attack helicopter missions. The attack helicopter application is probably the high-

est priority for ATR insertion. It is a good example of the functional capability desired. The mission is currently performed in the Army by the Apache helicopter with a crew of a pilot and weapons operator. The weapons operator, by viewing through an optical telescope, day TV, or thermal infrared imager, selects targets which are designated by a laser. Hellfire missiles fly to the laser spot on the target. The time taken by the operator searching for targets makes this platform vulnerable to enemy anti-aircraft fire. To shorten this time, in the new Commanche Helicopter, the operator will be assisted by an ATR process that possesses the sensor information and directs the operator to suspected targets.

The Navy's largest effort on detection and recognition focuses on undersea vessels using acoustics. This application is highly classified and does not overlap significantly with surface applications. Finding surface ships is done primarily with radar. There is a need for anti-ship missiles to autonomously attack the most valuable (dangerous) ships in enemy battle groups. This requires ATR capability. Both the Navy and the Air Force share the need for air-to-air fighter identification.

The Air Force is the primary service for conducting air strikes into enemy-held territory. This covers close air support, attacks on second echelon forces tens of kilometers from the battle, and deeper resupply, power plant, rail centers, or storage depots. The mobile targets like advancing tanks or hidden targets like SCUD's are the more challenging ATR applications. The fixed targets allow for sufficient time to prepare references and attack profiles that make the ATR task easier.

The main distinction between tactical and strategic is in the significance of the targets. In the ATR world, it is generally easier to automate the functions against the strategic targets than tactical ones because more resources can be devoted to the problem. Also, in tactical situations, the background is continuously changing because targets are generally mobile. The underlying processing technology for tactical, and strategic situations is quite similar.

It is to be noted that the effectiveness of autonomous smart weapons guided by multisensors that we have witnessed recently during Desert Storm was almost entirely against fixed targets. This allowed significant mission preplanning. To achieve comparable results against mobile targets will require a major infusion of ATR technology.

Because of the wide variety of missions, sensors, and functions which an ATR system may be required to deal with, there is no single solution to the algorithm or hardware task. For each application, the information available or attainable must be uniquely matched with the appropriate processing required for the functional needs of the mission.

3 Problems in Developing Real-World ATR Systems

The nature of the ATR problem is characterized by nonrepeatability of target signatures, competing cluttered objects, obscured targets, low contrast (for some sensors), long range (low resolution), conflicting evidence, natural variability, presence of camouflage, concealment and deception, and a wide variety of outdoor scenarios (geographic areas, weather and battlefield conditions), etc.

Some of the problems associated with developing ATR systems are:

- **Robust Algorithms**

  The key scientific problem is the absence of robust image understanding algorithms that can work in multiscenarios. Current image understanding algorithms do not provide the necessary consistency, reliability and predictability of results. For example, current algorithms result in high false alarm rates in images with high clutter and poor recognition performance in images without well defined target signatures. They make very little use of *a priori* information related to and present in the image, and generally make little use of meta-knowledge control.

- **Validation and Performance**

  In the current ATR systems, there is an absence of validation of data and models. Although we can measure some performance, we lack metrics to measure the input, i.e., useful techniques to characterize the input data are not available. Also there are problems associated with real-time performance evaluation. There is a need for suitable multiscenario databases. As a result, current experience in the ATR field is only with limited multisensor databases which are not very representative of scenario variability.

- **Lack of Specific Mission Requirements**

  Because the users of this technology do not have a clear idea of what capability is achievable, there is a reluctance to establish firm requirements. Instead, all the services have listed ATR as a desired capability for a variety of missions. A panel under the DoD Working Group on ATR Technology has listed these mission needs. However, the recent major shift in the perceived U.S. threat is likely to modify this.

- **Software/Hardware**

There are problems associated with specialized or general-purpose efficient processor architectures with hardware/software programmability and managing both numeric and symbolic computation. General purpose architectures are hard to optimize with respect to specialized processing and special purpose architectures have problems with flexibility and programmability. Also, there are no standard modules, hardware or software, to allow economy of scale to reduce costs.

- **Computational Power**

  Since the mission is not specifically defined, there is no simple answer to required computational throughput. It is clearly a function of the bandwidth of the incoming data, the functions to be performed, and the complexity of those functions. In most cases, the processing will be expanded to the capability of the available hardware. Today, this means that algorithms can be developed using billions of operations per second for applications requiring no more than a few small cards.

- **Man-machine Interfaces**

  Current target cueing technology can provide reasonable help in the field. However, the important research issues related with cueing are reliable target acquisition and man-machine interfaces for presenting the information to the operator/pilot. The interface concepts are just beginning to emerge about how to best use the operator and ATR together. What information, how to pass it and when to pass it, are some of the unresolved issues for most applications.

- **Technology Transfer**

  As a result of the proprietary nature of algorithms and systems, sharing of algorithms, software, hardware and data is difficult.

4 Status of Technology For ATR Systems

Significant progress has been made during the last ten years from adhoc techniques like looking for hot spots in infrared images by thresholding on contrast measures to more scientific understanding of sensors, algorithms, architectures, processors, systems and associated technology for software/hardware implementation [3].

During this time, the need for better databases used in training and testing ATR systems has been realized. Although more and better data will always be desired, several data sets have been established with ATR development in mind. In general, they have been characterized better than previous data sets and often have machine readable ground truth or image truth with them, rather than just hand scribbled log books.

The processor technology has been revolutionary. In 1980 an algorithm often took 30 minutes or more to run on a general purpose computer such as a VAX 11/750. To get the times down to a few seconds, in order to process a large number of images, dedicated processors were required for which algorithms could not be changed without modifying the hardware, a task often requiring several months. Today, commercial signal processing hardware exists to perform many of the component modules needed in ATR functions. This hardware is quite valuable in an R&D environment.

Simple algorithms to achieve real-time performance have been implemented in hardware and several such systems exist. Some of these systems have also been field tested. These systems are based purely on statistical pattern recognition algorithms. They employ limited multiframe processing analysis and possess no countermeasure capability. With inadequate training data, these algorithms alone could never hope to achieve the robustness required in many military applications. Only for limited low clutter and high contrast FLIR imagery, the detection and classification performance has been satisfactory.

At present, in laboratory prototypes or in simulation, ATR systems employing more complex algorithms and cross-sensor features including some model-based ATR are being developed. These systems use multisensors (i.e., FLIR, MMW, LADAR, SAR etc.) but make limited use of models, *a priori* information, countermeasure resistance, and scene analysis techniques. They have been tested on very small databases. Only for unobstructed medium contrast FLIR imagery, the recognition performance has been from marginal to satisfactory. Truly model-based multisensor ATR systems do not yet exist in real-time hardware systems.

There is now a larger set of "tools" in the algorithm developers "toolbox." They include knowledge-based tools, model-based tools, neural-nets, and genetic techniques. However, no one of these techniques, alone, is likely to be the solution to all ATR problems, but by applying the most useful techniques to each piece of the problem, progress is accelerating.

5 Challenging Image Understanding Problems

As discussed earlier, ATR is a system that involves sensors, algorithms and processors. The area where image understanding can significantly contribute is

in algorithm development. New, improved, robust algorithms will help to increase the effectiveness and usage of ATR systems. They will provide a better understanding of complex interaction between input data, models and output results. In the following we relate ATR problems to image understanding and identify potentially new research areas in image understanding that will help to solve the ATR problems.

- **Characterization of Input to an ATR System**

  An ATR system may be viewed as a vision system that consists of a variety of multisensors and multisource data. It is desired to characterize the input to an ATR system so that we can relate the inputs to the outputs of the system. This relationship is required for understanding the behavior of the system under a wide variety of inputs. This analytic or parametric relationship in turn also helps to provide prediction capability that is essential for practical uses of the system.

  We know a number of measures (such as probabilities of detection, false alarm, confusion matrix, etc.) that we can use to evaluate the output of the system. However, we do not know precisely how to characterize the input. A number of simple information measures such as edge points, entropy, uniformity, and structural measures have been proposed in the past. However, they are of limited value in characterizing the input objects and clutter. We need better measures that are realistic and may provide insight into the behavior. We would like to have some fundamental results (like in Information Theory) that set the bounds on the performance for recognition that is based on the information content in the sensor and other relevant data.

- **Consolidated Recognition and Motion Analysis**

  In a typical ATR system, tracking is generally a component of the system. Tracking is normally done by using a multimode tracker consisting of some combinations of 2-D feature matching, centroid matching and correlation matching. In the past, the 3-D information available in a sequence of images has not been used to improve tracking within an ATR system. Further, in ATR systems and image understanding, motion analysis has not been used in close cooperation with recognition to improve the recognition rates [4,5]. Also, very little work has been done where recognition helps to improve motion analysis.

  The idea is to consolidate recognition and motion analysis.

  (a) Use 3-D depth information to improve target tracking through occlusion and high clutter situations which in turn will improve recognition performance.

  (b) Integrate tracking and recognition functionalities, where one improves the performance of the other. As an example, in the long range detection of aircraft signatures, IR signature of an incoming threat target is detected using a highly stabilized, high sensitivity IR sensor and by separating true targets from the background. Such a sensor is called the IRST (InfraRed Search and Track). Using the IRST sensor, target identity and range is passively estimated using known motion of the sensor and assumed motion of detected incoming targets. The idea is to use something like IRST for ground or surface targets.

  (a) and (b) as described in the above can be developed independently or in conjunction with each other.

- **Model-based ATR**

  Model-based ATR is an extremely important research area and requires much advancement. In the IU community, mostly visible and some SAR and laser data has been used [6]. We need to develop expertise in other sensors (such as FLIR, SAR, MMW, etc.) which are so vital to the development of practical systems [9]. Further, the IU community has mainly dealt with geometric models of objects and limited (visible, SAR) sensor models. Many things are desired here.

  Model-based ATR involves developing not only the geometric models of the targets, but also models for sensors, clutter, background, atmospheric physics and countermeasures.

  A capability to generate and predict in real-time the signatures of targets under varying condition will be extremely valuable as are the needs for clutter and background modeling.

  Accurate and efficient development of sensor (e.g. thermal, SAR waveform, multispectral, laser, etc.) models and object models, and their use in model-based recognition and tracking, will be of importance in enhancing the recognition performance and reducing false alarms. It will also be desired to quantify the effect of sensor improvement with the performance of the system and validate various models used in the recognition process.

Developing models for algorithm behaviors relating input data/scene properties to algorithm behavior will be valuable. Also important are approximate model matching techniques in a dynamic environment which may be suited for explicit control, and fast, parallel implementation.

- **Robust Algorithms**

  "Robustness" as needed for ATR tasks is distinct from what it is perceived in the IU community. "Robustness" of an ATR system is the measure of insensitivity to the deviations in assumed input conditions to the system. One way to characterize "robustness" of the algorithms could be to relate the input information to the performance of the algorithms. A measure of "robustness" could be the number of false alarms for the same set of targets, but under different environmental conditions. The terms "robustness" and "multiscenario" are related. "Multiscenario" is implying "generality" of algorithms to work under varying environmental conditions in the outdoor scenario.

  Robust algorithms for segmentation and 3-D object recognition are desired. Since a target may give rise to a limitless variety of images (varying viewpoint and environmental conditions), it is important to use all the available knowledge and sensor information to accomplish the system goals.

- **Adaptive Algorithms**

  The single most important thing that will have one of the greatest impacts on ATR performance is the use of adaptive algorithms that can learn and adapt to the varying environmental conditions. No matter how sophisticated an algorithm may be, it will always have some imagery or conditions that will cause it to breakdown unless it has inherent learning capability. "Adaptiveness" ensures "robustness" as defined in the above. "Adaptive algorithms" are indeed "robust algorithms." However, adaptive algorithms may also possess self-calibration and learning capability which is not a fundamental requirement for robust algorithms. Adaptive algorithms are desired for segmentation, feature extraction and object recognition [2,7]. Techniques for rapidly training these subsystems are also desired.

- **Architectures for Integration of Auxiliary Information and Multisensors**

  Multisensor integration technology must be augmented with a priori information to provide improvement in target detection and classification performance. Use of auxiliary information like maps, navigational sensors, metrological data, altimeter, etc. into the recognition and tracking process will be extremely valuable in enhancing recognition and reducing false alarms. We need to develop flexible open system architecture that will allow integration of knowledge databases.

  (a) Integration - Mission-based, model-based integration of sensors, relevant data and information. Multisource information integration may include intelligence, mission planning, evidential reasoning, geography, doctrine, inflight updates, previous sightings, metrology, and models for target, clutter, countermeasures, sensors, etc. Integration process should allow for mission specifics so as to follow suitable recognition strategy based on available resources. All this will ensure the effectiveness of performance.

  (b) Knowledge databases - Good target recognition will not be achieved unless one updates knowledge databases dynamically so as to adapt to the changing situations. Currently, necessary software technology base is far behind. This makes the architecture use difficult.

- **Recognition and Guidance of Sensors**

  There are many mission scenarios where recognition and guidance of available sensors are closely related. In the IU community active vision and task-based vision paradigms have been popular for several years. For this technology to have an impact on the ATR problem, we need to apply this technology to the outdoor scenarios where the problem complexity is significantly higher than in the indoor scenarios.

- **Parallel Algorithms and VLSI**

  Parallel algorithms and mechanisms for mapping algorithms to specialized architectures need to be developed. Real-time digital/analog VLSI implementation of algorithms and subsystems/modules useful in various applications need to be developed. A lot of it can be done in a university/industry collaboration by developing and expanding working relationships between industry and universities with support from the government.

- **Real-time Performance Evaluation**

  We need to develop the framework for building vision systems so that real-time evaluation and performance characterization of algorithms can be done in laboratory prototypes or in actual field tests. This framework needs to be common among various sites so pieces that are use-

ful from various researchers can be integrated together.

The availability of some multisensor, registered database of image and associated ground truth to IU researchers will be very valuable. Standard data sets for various applications can help focus the research efforts, allow comparisons and expand the expertise of the IU community to ATR problems.

- **Man-machine Interfaces**

  Suitable interfaces need to be developed to apply the image understanding and human factors technology to the "person in the loop" situations.

6 Conclusions

The development of automatic target recognition systems that can perform under dynamic environmental conditions will be of great practical significance in many DoD applications such as reducing workloads of pilots and tank commanders. We hope that the solution of image understanding problems as discussed in this paper will lead to further discussions of image understanding research for ATR application. All this will result in scientific advancements in the IU field and more effective and robust ATR systems in the future. The technology developed will also be applicable to other areas of IU applications such as navigation, photointerpretation and robot vision.

7 Acknowledgements

The authors would like to thank Ram Nevatia, Takeo Kanade, Subhodev Das and Matt Barth for useful discussions.

References

[1] B. Bhanu. Automatic Target Recognition: State of the Art Survey. *IEEE Trans. on Aerospace & Electronic Systems*, Vol. AES-22, No. 4, pp. 364-379, July 1986.

[2] B. Bhanu, S. Lee and J. Ming. Self-Optimizing Image Segmentation System Using a Genetic Algorithm *Proc. Fourth International Conference on Genetic Algorithms*, San Diego, CA, Plenary Presentation, pp. 362-369 July 1991.

[3] S. Chow and T. Jones. Army's FLIR/ATR Evolution Path. *SPIE*, March 1989.

[4] A. Hanson and R. Kumar. The Application of 3-D Pose Determination Techniques to 3-D Model Extension *IEEE Special Workshop on Passive Ranging*, Princeton, NJ, October 1991.

[5] T. Huang. 3-D Motion Estimation and Object Recognition from Stereo Image Sequences. *IEEE Special Workshop on Passive Ranging*, Princeton, NJ, October 1991.

[6] K. Sato, K. Ikeuchi and T. Kanade. Model-based Recognition of Specular Objects Using Sensor Models *IEEE Workshop on Directions in Automated CAD-Based Vision* Maui, Hawaii, pp. 2-10 June 1991.

[7] J. Ming and B. Bhanu. A Multistrategy Learning Approach for Target Model Recognition, Acquisition and Refinement *Proc. DARPA Image Understanding Workshop*, pp. 742-756 September 1990.

[8] Infrared Systems and Components. *SPIE*, Vol. 750, 1987.

[9] E.G. Zelnio. The Importance of Sensors Models to Model-Based Vision Applications. *WRDC/AARA paper*, 1989.

ATR: Problems and Possibilities for the IU Community

Jan D. Wald
David B. Krig

Alliant Techsystems, Inc.
600 2nd Street N.E.
Hopkins, MN 55343

Trent DePersia

Advanced System Technology
DARPA
Rosslyn, VA

Abstract

Because of poor communications between Image Understanding (IU) and Automatic Target Recognition (ATR) communities, IU research has had a negligible impact on ATR developers. In this paper we will provide some possible applications of IU technology that can assist in ATR problem solutions. First, we define the ***Current ATR Dilemma*** *and what makes the ATR problem a difficult challenge. Following this discussion, we highlight* ***ATR Research Challenges*** *and present applicable IU research.*

1. THE CURRENT ATR DILEMMA

Although Automatic Target Recognition (ATR) systems must function in an extremely noisy environment, the tools, techniques, and methods available to them work only in fairly sterile or well bounded domains. Recently, model based approaches have been used to improve performance. Even using these methods, rigidly constructed models do not produce the flexibility or relaxation methods for true multiscenario recognition.

In the past ATR developers believed that the primary ATR performance limitations were *real-time hardware, sensors performance,* and *the availability of data.* This assumption is no longer true. Commercial hardware is now available. The belief that sensor resolutions do not allow internal target or backgrounds details to be seen is also untrue. New sensors can provide fine grained target and environment information that not only make newer approaches feasible but necessary as well. Blob detection is no longer the key to target recognition; gathering clues of target identity and reaching an educated conclusion is now practical and central to future improvements in ATR performance. Data availability is still an issue. Data collections are expensive and are often classified, which limits their availability. But enough data exists that if needed a sufficient data base can be developed for meaningful ATR development programs.

Today, the challenge is to show that collaboration between the ATR developers and the IU community is mutually beneficial.

- *ATR research and development has much to gain from teaming with the IU community*. Many of the promising ATR approaches have been derived indirectly from the IU community (graduate students' movement into industry).

- *The IU community can gain equally from exposure to the challenges facing ATR developers.* Just as the ALV application drove research in the 1980's, the ATR application can drive research in the 1990's.

These challenges are open opportunities. The criticality of the goal was amply demonstrated in Desert Storm. An early success in moving research to a fieldable end product for ATRs will justify the IU program as few other arguments will.

1.1 ATR Applications

ATRs have many uses. In this section we will describe their proposed applications and the hostile environment in which ATRs are being developed to operate.

All vision problems are hard, but not all are hard for the same reason. When looking at Vision Research as a scientific endeavor, domains can be simplified so that the essence of the problem being investigated can be isolated and studied. Experimenting and developing the means to represent and capture meaningful information offers unending opportunities for a researcher—even in the simplified domain that is being investigated. The results can often be appropriated for other applications where the simplifying assumptions made may be directly transferred into the domain. However, this is not the case in ATR applications.

If the domain itself is unconstrained, then the research has to focus not so much on representation, but on control. The fundamental challenge is to determine how a vision system should function in order to handle the variability inherent in complex, outdoor contexts, not for relatively simple tasks as Landmark Recognition, but for problems such as recognizing concealed relocatable high valued targets. Until now solutions have been constrained because of the lack of adequate hardware; so, IU solutions would have been impractical. Sensor resolutions have not permitted IU exploitation. Today hardware and resolution are no longer problems. ***So the challenge now is to develop the techniques that can exploit features found in ATR application domains.***

ATR applications can be described as:

- *Penetrator/Missile*—Shoot-to-kill, move-to-kill, move-to-position, search, and kill;-
- *Land Vehicle*—RSTA, direct fire, indirect fire, surveillance, reconnaissance, perimeter defense
- *Air Vehicle*—Cueing, indirect fire, surveillance;
- *Space Borne*—Surveillance, reconnaissance.

While each of these applications has its own set of challenges, all must deal with a set of general conditions:

- *Scene Contents*—Stationary targets, moving targets, one-on-few, one-on-many
- *Target Types*—Clutter conditions, likely ranges
- *Scenario Variables*—Weather conditions, day/night, varying terrain;
- *Battlefield Conditions*—Countermeasures, dust/smoke/fires, occlusion;

1.2 Smart Weapons Application

Smart weapons will be used to conduct operational missions too difficult and/or dangerous for manned systems in battle arenas such as Close Air Support, Battlefield Air Interdiction, Deep Strike, Offensive Counter Air, Defensive Counter Air, Surveillance/Sigint/Comint and Communications. Their primary mission is to:

1. Seek out enemy threats
2. Select appropriate targets
3. Report/destroy them.

Moreover, smart weapons will have to perform these tasks in a cluttered environment with dust and other environmental factors affecting performance, where the target or threat is actively

concealed. It will have to be able to move into an area, search it to find the threats/targets, select an appropriate one if necessary and then destroy it. *Stealth is important; planning is important; but, **sensing, situation awareness, and searching are critical.***

Two kinds of platforms are typically employed, an Autonomous Air Vehicle for long hauling and searching and an Intelligent Munition for final targeting and delivery. Typically, multisensor suites are used containing MMW, LADAR, and FLIR sensors. Multiple sensors are needed because multiple bandwidths and active systems provide ranging and spectral information, under varying scene and atmospheric conditions, that increase the probability that the mission will succeed.

2. RESEARCH CHALLENGES

Primarily, ATR research challenges are concerned with finding ways to identify and classify targets under various environmental and scene conditions. Different sensory modalities offer different advantages and disadvantages. The task usually determines which sensor or specific combination of sensors is used.

2.1 Target Signature Variations

Target signature variations in FLIR and radar data arise from the target being in different states. The different states are produced by varying conditions in the scene. For FLIR imagery, the influence of the atmospheric effects is not well understood. Some of the variables are:

- Vehicle Orientation Variables—
  - Aspect angle
  - Image plane rotation
  - Depression angle
- Vehicle Thermal Variables—
  - Engine (running / not running)
  - Vehicle motion (stationary / exercised)
- Environmental Variables—
  - Solar loading (nighttime / daytime)
  - Sun angle
  - Wind
  - Obscuration (fog / rain / dust / smoke)

2.1.1 FLIR Signature Variations

Heated objects emit light having a spectrum characteristic of their temperature. In a narrow range of wavelengths in the far infrared region of the spectrum, the temperature of an object can be determined by measuring the intensity of that light. Hotter objects emit higher intensity than cooler objects. This is the basis for infrared sensors.

In Forward Looking Infrared (FLIR) imagery, tactical vehicles such as tanks, APCs, and trucks usually appear brighter than their surrounding background due to the greater thermal emission from the targets compared with the background. In FLIR imagery, the contrast is caused by differences in temperatures rather than differences in intensities of reflected light as in visible band imagery.

A new generation of infrared sensors having higher spatial resolution and greater thermal sensitivity has been developed. Second generation FLIR sensor image quality is significantly better than that of the previous generation FLIR sensors. A target's structural details are clearly visible at longer ranges and with higher fidelity. Object recognition algorithms developed within the IU community for visible imagery are of great interest within the ATR community. As in the case of visible imagery, an object's appearance varies greatly depending upon several object and environmental factors.

The appearances or signatures of tactical vehicles in FLIR are highly variable depending on factors attributable to the vehicle, the sun, and the atmosphere. In addition to thermal emissions from the targets, there are also emissions from the surrounding background caused by reradiated heat.

Figure 1 shows examples of three different targets in two signature states. In the upper row, all of the targets have their engines running and are exercised and their FLIR signatures are bright in the locations of their engines and exhausts caused by thermal conduction from engine combustion and their wheels and treads caused by frictional heating.

In the lower row, the engines are not running and so there is no thermal signature from the engine. The targets are not exercised so there is no frictional heating from the wheels or treads.

Figure 2 shows the effect of aspect angle variations on a target's appearance in FLIR. This figure shows six views of an M35 truck. The engine is visible in all of the front facing aspect angles while it is not in the rear facing aspects, making it more difficult to recognize. Notice how the position of the exhaust pipe is a strong clue to the vehicle's aspect angle.

ATR research has focused on solving the recognition problem for all of the different vehicle orientation and thermal variables. However, because of the difficulty in solving this constrained class of the problem, less emphasis has been placed on solutions involving variations of the environmental variables.

FLIR Modeling Software

Infrared modeling research has been conducted for many years towards understanding the impacts on both targets and backgrounds of the state variables listed above. An example of recent progress in FLIR modeling for targets is PRISM[1]. PRISM incorporates geometrical models of vehicles with efficient approximations to the solutions of first principles heat transfer equations. It is an effective tool for visualizing target signature variations given the a priori values of many target and environmental variables. Research is continuing toward accurately predicting the signatures of targets in simulations of complex real-world backgrounds.



Figure 1. Examples of Signature Variation. a.) M60 tank with hot engine and treads, b.) M60 tank with cold engine and treads, c.) M113 APC with hot engine and hatch, d.) M113 APC with hot engine but cold hatch, e.) M35 truck with hot engine, exhaust, and bed, f.) M35 truck with hot engine and exhaust but cold bed.



Figure 2. Six Aspect Angles of an M35 Truck Showing its Changing FLIR Signature as a Function of Aspect.

2.1.2 Radar Signature Variations

In the Millimeter Wavelength (MMW) region of the electromagnetic spectrum, radar systems have been designed that have a narrower beamwidth than the more prevalent tracking radars that operate in longer wavelength bands. Higher spatial resolution is obtained by using MMW radars making it possible to resolve tactical targets at ranges out to several kilometers. Radar systems have recently been developed that employ MMW technology and provide signal data in the forms of high range resolution profiles, spatial super resolution returns, polarimetric data, and Doppler. In all of these cases, the radar pulses are reflected by targets and clutter.

Reflections from the discrete scatterers that make up the high range resolution signal have a very narrow reflection peak. The narrow width of the peak is caused by the nature of the reflection, mainly specular, dihedral, and trihedral; all of these have very narrow peaks.

Figure 3 shows two plots of MMW radar HRR profiles from an M60 tank. The axes represent the range in meters and the aspect angle in number of looks at the target where looks are spaced approximately 0.02 degrees apart. The aspect angle varies only by 1.1 degrees over the plots. The prominent peaks in the signal appear for only about one look or 0.02 degrees and then become less prominent, yielding to other new peaks. This behavior is consistently observed in HRR profile plots. Unfortunately, this behavior makes the recognition problem extremely difficult.

Figure 4 shows the length feature as a function of target aspect angle. The length feature is extracted from an HRR profile by locating the first and last bins in the profile. In this figure, each aspect bin is a histogram of the target length for that aspect angle computed over many looks at the target. The length feature is quite noisy. There appears to be a dominant length for each aspect that when traced connected over all aspects, it traces out the characteristic aspect angle signature of the target.

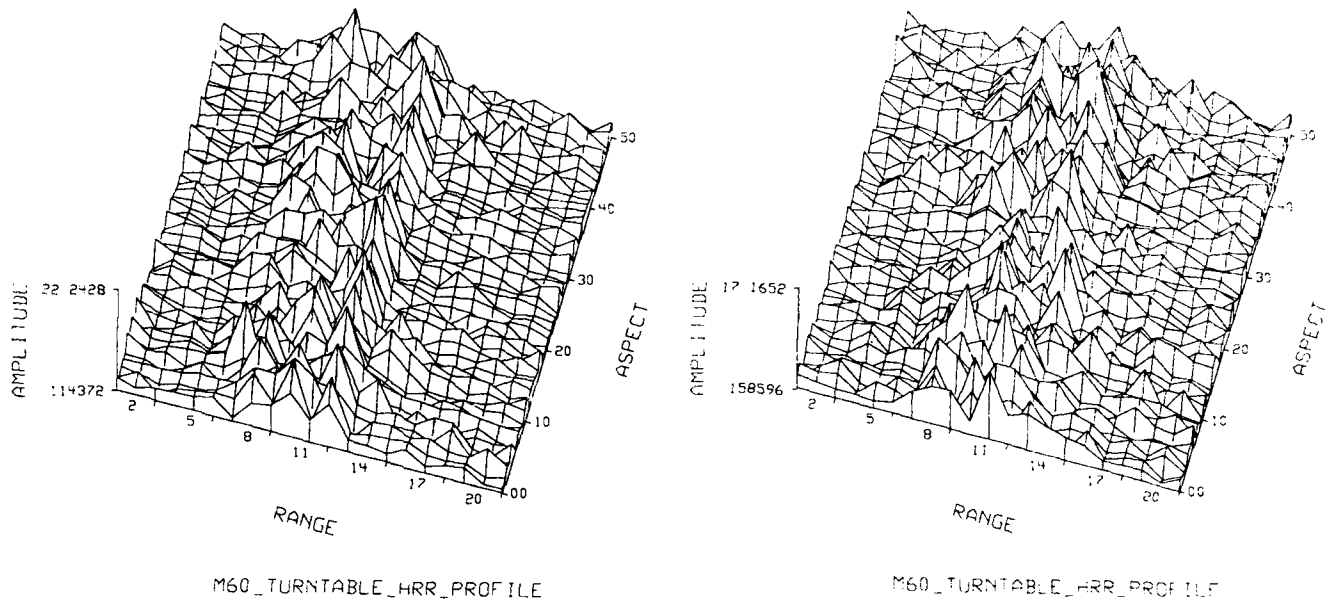


Figure 3. MMW Radar HRR Plots for a Tank. Range axis is in meters while the aspect axis is in looks separated by approximately 0.02 degrees in aspect for a total aspect change of 1.1 degrees. Notice how irregular the peaks are from look to look.

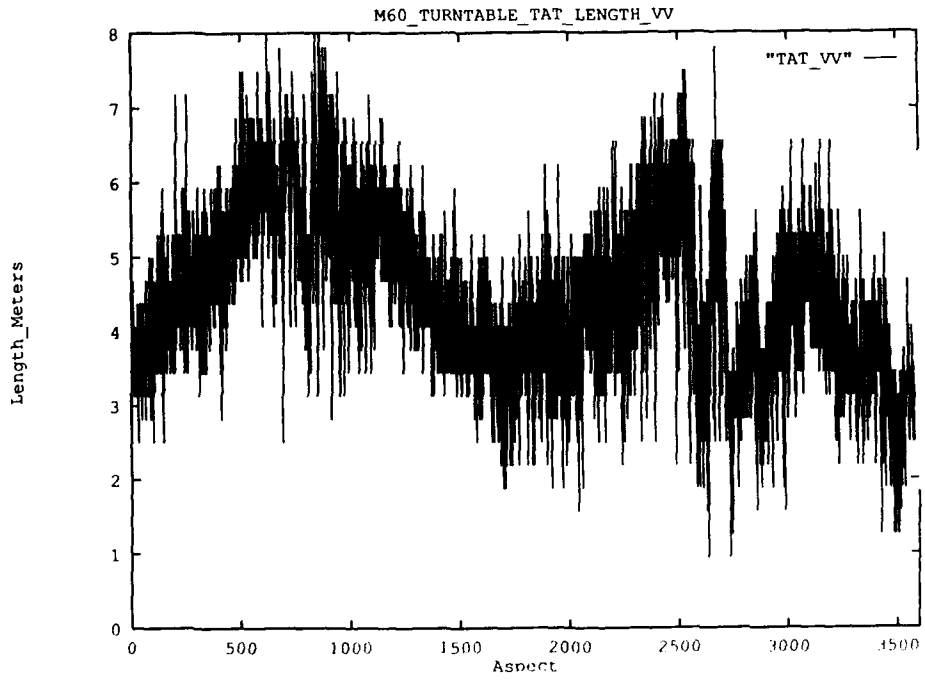


Figure 4. Histograms of the Length Feature Extracted from HRR Profiles for a Tank. Each aspect angle bin is a histogram of the length feature from numerous looks at the target at the associated aspect. Although there is a trend visible over all aspect angles, each aspect bin exhibits a large amount of uncertainty.

Radar Modeling Software

Research for predicting accurate radar signatures from geometrical models of vehicles has progressed recently. One example is the TRAK radar modeling package developed by Georgia Tech Research Institute[2]. TRAK couples

multimode, multiband radar modeling with geometrical vehicle models built using the MAX Geometrical modeler[3].

2.2 Target Occlusions

The most frequently encountered occlusions occur when terrain between a sensor and target occludes a target. This type of occlusion is of concern in ground-to-ground scenarios (both the sensors and the target are on the ground). Other types of occlusion that occur in air-to-ground scenarios are caused by other man-made or natural objects. Trees, rocks, buildings, and other targets are occluding objects. Neither the percentage of the target that is occluded nor the part of it are known a priori thereby, complicating recognition.

2.2.1 Dynamic Occlusions

Dynamic occlusions occur whenever the occlusion changes over time. The situations in which this happens are:

1. *A moving target moves behind stationary objects or another moving target*

2. *A moving target occludes a stationary target.*

Dynamic occlusions begin with an unoccluded target becoming progressively more obscured over several frame times until a point of maximum occlusion after which the target becomes progressively less obscured until it is once again unoccluded. It is reasonable to assume that since the target will reappear in a short time, it is unnecessary to continually recognize the target through the occlusion. Rather, a target tracking algorithm might be employed to coast the target state through the occlusion and then continue performing recognition once the occlusion is over. Dynamic occlusion is not as serious as the static occlusion.

2.2.2 Static Occlusions

Static occlusions do not change over time because there is no relative motion between target and occluding object or terrain. In this situation, recognition must be performed using only the visible portion of the target. This is perhaps one of the most compelling justifications for using detailed target models to perform recognition tasks. Since the part of the target that is occluded cannot always be predicted, the recognition algorithm should be able to use whatever portion is unoccluded to recognize the target.

2.3 Sensor Fusion

The generic benefits of fusing information from sensors of different type are well known; developers can take advantage of the specific qualities of each type of sensor to achieve performance that is superior to that of each individual sensor. The challenge for the ATR developer is to determine *when to fuse and at what level in the processing hierarchy.*

Sensor fusion has not been a primary focus of IU research. However, examples of it have recently been published by researchers at UPENN, CMU, and SRI. UPENN, for example, has completed some elegant work on the mathematical foundations of fusion[4,5,6]. Work at CMU in sensor fusion has focused on fusing 3D data from a LADAR with color imagery for vision aided navigation of the NAVLAB[7,8]. SRI has used fusion techniques to good purpose in their work on stereo[9].

In the ATR community, sensor fusion is performed at several distinct levels in the processing hierarchy: the *data association level*, the *information fusion level*, and the *decision level.*

At the *data association level*, the association problem is due to the vastly different spatial resolutions of radar and FLIR sensors. Radar spatial resolution is approximately an order of magnitude lower than that of FLIR. A symptom of this problem occurs whenever targets are closely spaced such that a single radar return contains reflected energy from multiple targets while the targets are resolved by the FLIR. The association problem for sensors having differing spatial resolutions has not been dealt with in the IU community.

At the *information fusion level*, information from each of the sensors has already been extracted and is to be fused in order to form hybrid information that can aid in improving

the discrimination of recognition algorithms. Figure 5 is an example of fusion at this level.

Two sets of radar plots are shown along with two sets of FLIR images. The radar plots look very similar and yet they are correlated to two very different FLIR images. On the other hand two similar FLIR images correlate to two very different radar plots. Supporting radar recognition with FLIR information would help distinguish the two targets. In the second case the radar would help determine whether the two targets recognized differed only in aspect angle. The problem is not how to fuse the information if the methods are well known. What is not understood is when to fuse the information given the situational context and at what level (*feature level, recognition level, detection level,* etc.)

At the *decision level*, the benefits from fusion come from fusing interpretations of the data from the radar and FLIR sensors using associated confidences. However, this level of fusion has traditionally required knowledge of a priori probabilities for the target classes and class conditional densities that are empirically derived. The way a developer makes fusion algorithms work is by tuning (tweaking) them until they achieve the desired results. Success is specific to the particular data set used and the ability of the developer to make enough of the right parameters available to tune.

ATR developers have been relying more heavily on models both to alleviate the problem of not having enough development data available and also to gain detailed understanding of the problem.

2.4 Articulated Targets

Tactical targets consist of moving parts, some of which are large enough to cause a change in the target's appearance when they are moved. Tank turrets are a simple example; but others are trailers towed by trucks and missiles mounted on missile launchers. In the case of articulation, the component of a target that could be rotated is known a priori, but the amount of rotation is not known.

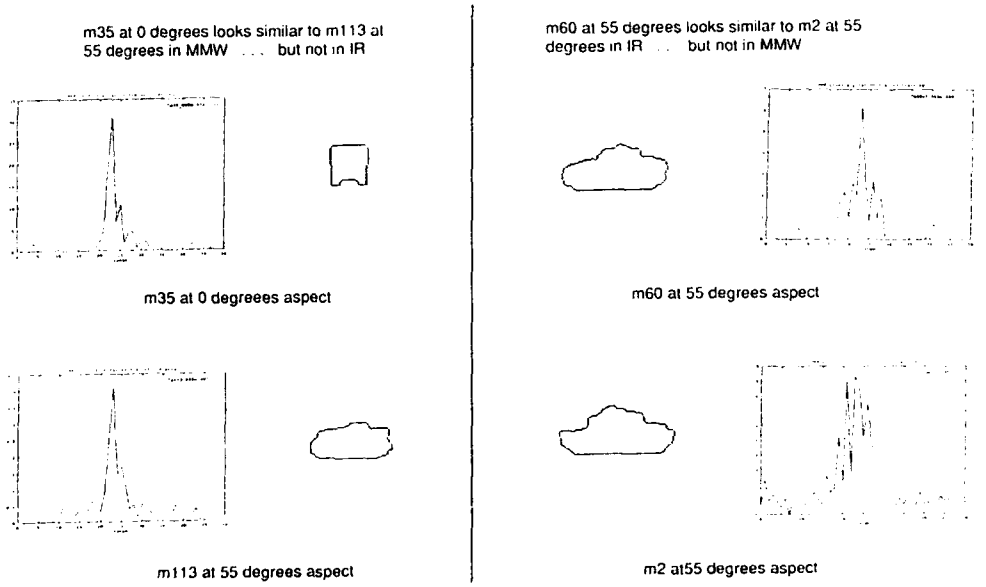


Figure 5. Examples Showing the Potential Benefits of Sensor Fusion. a.) The two MMW radar HRR plots are ambiguous but the FLIR images of the targets are quite different indicating that the FLIR image helps disambiguate the targets. b.) The FLIR images are ambiguous while their MMW radar HRR plots are quite different indicating that the radar data helps disambiguate the targets.

Figure 6 shows an example of two tanks with and without rotated turrets. Tanks are often seen in motion with their turrets facing the rear. This is done so that the driver has adequate clearance to escape from the drivers compartment in case of an emergency. Tanks are also sometimes observed in this configuration while parked.

As in situations involving occlusions, it is important to make use of partial target models that predict the possibility of articulated component rotation.



Figure 6. Two Tanks With and Without Rotated Turrets.

3. A CHALLENGE FOR IU

3.1 Target Signature Variations

The target signature variations derive from different objects and require different models for their recognition. The augmented model base approach increases the amount of search needed for recognition based on the number of signature variations. By treating the problem in this fashion, very efficient search strategies are needed possibly involving precompilation of object models into efficient search strategies. The IU community has been active in this study for the past several years and could have a positive impact on ATR research in this area.

Alternatively, signature variation derive from instances of the same object under different illuminations. In this case, there is a need to exploit the properties of the object that are invariant to illumination or thermal signature in the case of FLIR. The IU community has also done research in this area as well.

Choosing an approach depends upon the degree of ambiguity between objects in the model base, the size of the model base, and the search complexity and time constraints placed on the algorithm by the processor.

3.2 Occlusions

The IU community has focused on the problem of occluded object recognition. Many approaches have been developed that make use of partial object model matching to occluded object data. Some examples are:

- *The recent work on perceptual grouping of image features prior to model matching*[10]
- *Searching the interpretation tree with null model matches*[11]
- *The use of Hough transforms and geometric hashing techniques to limit the number of possible object hypotheses.*[12,13]

3.3 Sensor Fusion

Where IU can be beneficial to ATR is to develop general methods for determining which information to fuse (i.e., when to believe one sensor over another) and when to fuse it (i.e. does it help to add MMW information if it is raining). The overall problem is accounting for context. *The specific problem is in determining what context an ATR is in and given that it is there, to determine what can be believed and what can not.*

3.4 Articulated Targets

Articulated targets present a difficult problem because of the unknown relative orientation

of the separated parts. The problem is made more complex by the possibility of multiple articulated objects in the same model base and unconstrained view angles. The more recent work on perceptual grouping and geometric hashing may prove useful in providing solutions to this problem. Also needed are geometric reasoning approaches for hypothesizing instances of the known articulated objects in particular relative orientations given the partial descriptions from the grouped data.

4. CONCLUSION

An IU program devoted to ATR challenges promises significant benefit to DoD. The issues raised in this paper are by no means exhaustive representing only a quick look at the kinds of challenges for and applications of IU techniques in the ATR domain. A more thorough look would no doubt produce a longer and, perhaps, more interesting list of topics.

---

[1]PRISM User's Manual, Michigan Technological University Keweenaw Research Center.

[2]J. L. Davis, G. J. Bradley, R. B. Rakes, J. H. Andrews, M. T. Tuley, P. A. Ryan, *"TRAK 4.1 Radar Cross Section and Tracking Simulation User's Guide,"* July 1990.

[3]J. W. Peifer, R. B. Rakes, M. S. West, J. H. Andrews, MAX Geometric Data Base Editor User's Manual, Georgia Tech Research Corporation, 1988.

[4]Raymond McKendall, and Max Mintz, *"Non-Monotonic Decision Rules for Sensor Fusion,"* Proc. DARPA IU Workshop, pp. 874-880, 1990

[5]Raymond McKendall, *"Statistical Decision Theory for Sensor Fusion,"* Proc. DARPA IU Workshop, pp. 861-866, 1990

[6]Gerda Kamberova, and Max Mintz, *"Robust Multi-Sensor Fusion: A Decision-Theoretic Approach,"* Proc. DARPA IU Workshop, pp. 867-873, 1990

[7]Martial Hebert, "Building and Navigating Maps of Road Scenes Using Active Range and Reflectance Data," in Vision and Navigation: The Carnegie Mellon Navlab, ed. Charles E. Thorpe, pp. 117-129, Kluwer Academic Publishers, 1990.

[8]Martial Hebert, InSo Kweon, and Takeo Kanade, "3-D Vision Techniques for Autonomous Vehicles in Vision and Navigation: The Carnegie Mellon Navlab, ed. Charles E. Thorpe, pp. 130-186, Kluwer Academic Publishers, 1990.

[9]Stephen T. Barnard, "Stochastic Stereo Matching over Scale," International Journal of Computer Vision 3, pp. 17-32, 1989.

[10]David Lowe, *"Three Dimensional Object Recognition from Single Two Dimensional Images,"* Artificial Intelligence 31 (1987) pp. 355-395.

[11]W. E. L. Grimson, *"Object Recognition by Computer: The Role of Geometric Constraints,"* 1990, MIT Press.

[12]Dong Yoon Kim, J. John Kim, Peter Meer, Doron Mintz, and Azriel Rosenfeld, *"Robust Computer Vision: A Least Median of Squares Based Approach,"* Proc. DARPA IU Workshop, pp. 1117-1134, 1989.

[13]Rakesh Kumar, and Allen Hanson, "Analysis of Different Robust Methods for Pose Refinement," Proc. of IEEE Workshop on Interpretation of 3D Scenes, Austin, TX, pp. 52-60.

Building a Cartographic Infra-Structure: Simulation and Cartography

David M. McKeown, Jr.
Digital Mapping Laboratory
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

George E. Lukes
Autonomous Technologies Division
Research Institute
U.S. Army Topographic Engineering Center
Fort Belvoir, VA 22060-5546

Abstract

Emerging requirements for detailed spatial databases to support a variety of Department of Defense (DoD) activities may require substantial investment in basic research to advance the state-of-the-art in automated cartographic database construction and maintenance. This paper provides a technical rationale for pursuing a new initiative focused on the application of image understanding technology to automated cartography in support of programs for simulation and training, mission planning, autonomous navigation, and the analysis of intelligence imagery.[1]

1. Introduction

The goal of this proposed initiative is to provide a research focus in automated mapping and the interpretation of remotely sensed imagery to meet the needs of existing and emerging Defense programs. Typically, each program has a component that relies on the generation and maintenance of accurate spatial data to support decision-making by human military personnel and/or intelligent autonomous agents. A concerted effort focused in this area (akin to the Strategic Computing Program initiatives) has long-term relevance to several on-going DoD and Defense Advanced Research Projects Agency (DARPA) programs. These include major programs for advanced distributed simulation such as Battlefield Distributed Simulation (BDS-D) and Close Combat Tactical Trainer (CCTT), high-fidelity training simulators represented by the Special Operations Forces Aircrew Training System (SOF-ATS), Unmanned Ground Vehicles (UGV), and Research and Development for Imagery Understanding Systems (RADIUS). The spatial data requirements for these programs generally include significant augmentation of standard products produced by the Defense Mapping Agency (DMA) to address critical issues of timeliness, local geographic intensification, and operational security. They currently rely largely on manual and interactive compilation, which are labor-intensive and not well-suited to demands of responsive database intensification and maintenance. Application-specific enhancements are seldom shared with other users. Moreover, the emergence of mature networking technology is generating high-level interest in achieving interoperability between highly heterogeneous systems that require consistent terrain databases at widely varying levels of detail. Achieving and maintaining consistency between such diverse terrain databases is a major research and operational issue. Thus, there is a need to advance the state-of-the-art in the representation, construction, intensification, and maintenance of spatial databases to support a diverse and growing set of DoD customers.

[1] This work was primarily sponsored by the U.S. Army Topographic Engineering Center under Contract DACA76-91-C-0014 and partially supported by the Avionics Lab, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Topographic Engineering Center, or the Defense Advanced Research Projects Agency, or of the United States Government.

1.1. Cartographic Technology 'Backbone'

DARPA can take the lead in developing a responsive DoD cartographic technology 'backbone' to address those areas where static standard mapping products alone can not provide either the appropriate level-of-detail or timeliness. Currently, specialized DoD programs with requirements that can not be fully met by off-the-shelf DMA mapping products typically pursue generation of program-specific terrain databases that are neither timely, inexpensive, standardized, nor shared. Col. David F. Maune, Commander and Director of the U.S. Army Topographic Engineering Center (TEC), states the problem succinctly:

"Currently, the purchase price for each simulation system procured by the Department of Defense (DoD) includes the cost of developing its own unique data base, or the cost of transforming data bases developed by the Defense Mapping Agency (DMA) or others. This results in the proliferation of nonstandard, incompatible data bases and software which is costly for the government in terms of both recurring development and maintenance costs. In many programs, this also means that costly data bases are only produced for unique training scenarios and would not be applicable for rehearsals of actual missions on different pieces of real estate. We learned in Operation Desert Storm that mission rehearsal capabilities on the well-mapped and digitized terrain of the Fulda Gap in Germany didn't do much good in Kuwait."

Similar observations can be made regarding unique databases developed for various wargaming, mission planning, command and control, and intelligence applications.

Several initiatives are underway to develop spatial data exchange specifications (e.g., DMA's Vector Product Format, Air Force Project 2851s SSDB Interchange Format or SIF) which promise improved mechanisms for sharing extracted spatial data. These provide a framework for exploring the technical and programmatic issues associated with generating and maintaining consistent terrain databases in diverse real-time formats and at varying levels-of-detail.

Our technology development proposal should be viewed as addressing the gap in the generation of specialized and highly timely spatial data, not as solving the DoD mapping problem in the large. There are several general classes of high visibility DoD programs that could benefit from an enhanced technical base to support the production and maintenance of cartographic data. These include:

1. Simulation, mission planning and rehearsal, training, and image perspective transformation (IPT).
2. Advanced weapon systems involving autonomous guidance, highly accurate terminal homing, and autonomous target detection (smart and brilliant weapons).
3. Unmanned air and ground vehicles (UAV/UGV) employing autonomous navigation technologies.
4. Aids for the analysis of tactical and national imagery including basic site mapping and terrain modeling.

2. Technology Development Areas

We have identified four technology development areas that provide a rich set of basic and applied research problems, represent areas that are currently under emphasized in the DARPA IU research program, and will provide leverage for and synergy with ongoing DoD development and DARPA research programs. These areas are:

2.1. Spatial database intensification

Given a 'standard' digital map product such as Interim Terrain Data (ITD) and high resolution aerial photography, investigate automated techniques to improve the level-of-detail by detection, delineation, and attribution of man-made objects such as buildings and roads whose scale precluded their compilation into the original digital map. Integrate the extracted spatial information into the original 'standard' digital map product format to illustrate the concept of a seamless database update.

2.2. Hierarchical generalization of spatial data

Current trends in heterogeneous spatial data to support mission rehearsal, simulation, and IPT require data at a variety of spatial resolutions and accuracies. Given a standard digital map product, such as ITD, aggregate and generalize the spatial data by well defined criteria to generate a set of corresponding digital maps at successively coarser levels of resolution. Support map update by propagating changes in each level of representation. Correlated representation of geographic areas at multiple levels of detail are required to support integration of diverse models, simulations, and simulators in large heterogeneous distributed systems.

2.3. Information fusion from multiple data sources

For the most part, current research in computer vision for cartographic feature extraction and photo-interpretation has relied on digitized aerial photography using panchromatic imagery. The mapping community has shown the utility to produce large-scale photographic map products using Landsat TM and SPOT satellite imagery. There exists a body of baseline digital elevation and coarse resolution map data (DMA DLMS, USGS DLG, etc.). A research initiative that looks for applications that emphasize the integration and fusion of multiple data sources would bring current IU/AI technology to bear in an area which heretofore has received little attention by the research community.

2.4. High resolution multi-spectral image analysis

Over the last twenty years, the remote sensing community has focused its efforts on the analysis of remotely sensed multi-spectral imagery such as Landsat MSS, Landsat TM, and SPOT using computational models derived from statistical analysis. As the pixel size has shrunk from 80 meters, to 30 meters, to 10-20 meters, the opportunity has increased to apply structural and spatial analysis techniques developed for high resolution aerial imagery to satellite data. Further, a new generation of high resolution airborne multi-spectral scanners (Daedaleus, AVRIS, MEIS, etc.) can be flown to support multi-spectral pixel resolutions well below five meters. An opportunity exists to develop IU technology to support the automated analysis of high resolution multi-spectral imagery that goes beyond traditional statistical analysis. The variety of applications and potential impact for automated surface material classification, more accurate map feature attribution, and improved thematic and land-use maps makes this an important technology development area.

3. Project Approach and Technology Transfer

A more substantive study should be tasked to produce detailed descriptions of key application problems. This should be initiated as soon as possible. Once existing image understanding (IU) technology areas have been matched with particular problem domains, the goal is to show the applicability of IU/AI techniques to those problems via image analysis and spatial database construction tasks. This will require cooperation between IU researchers, developers, and the end-user community to assemble the variety of imagery, map, and terrain data necessary to support detailed experimentation and evaluation. It is clear that the current IU technical base in stereo matching, monocular image analysis, three-dimensional modeling, information fusion, etc. provides the technical foundation for this project. What is needed is a focused and concerted effort for improved performance across more complex task domains containing a variety of source imagery, a range of scene complexities, and guided by human-level performance metrics arising from the requirements of the end-user community.

Technology transfer will be accomplished in part by milestone demonstrations in government pilot plant facilities such as the TEC Terrain Analysis Center, the National Exploitation Laboratory (NEL), and various DMA techniques groups. Such demonstrations serve to show proof of principle on representative operational datasets.

3.1. Potential Customers

As previously discussed there are many potential customers with DoD for improved techniques to build detailed cartographic databases to support simulation, intelligence analysis, mission rehearsal and planning. We focus on two illustrative programs, one within DARPA, and the other within the Air Force.

DARPA is starting a diverse, multi-office program in Advanced Simulation built on the distributed simulation technology pioneered in the highly successful DARPA/Army SIMNET program. One thrust of the program is to expand the original focus of armor unit training to include all service training exercises at progressively higher echelons. Proof-of-concept was initially demonstrated by incorporating close air support and air defense artillery to augment the armor maneuver force in the simulation network. It was then extended to include naval forces with coordinated off-shore naval gunfire and cruise missile capabilities demonstrated by DARPA and the Navy in Battle Fleet In-Port Training (BFIT) exercises.

Advanced Simulation surfaces many open issues in the representation of digital terrain data to provide levels-of-detail ranging from highly efficient formats to support fast-moving strategic and tactical aircraft down to fine-grained terrain databases for dismounted infantry. During the course of a battle simulation the terrain may be modified due to battlefield activity requiring the instantiation of modified terrain representations throughout a network of heterogeneous

simulators. Issues in temporal representation of micro terrain features may have bearing on techniques to extract and maintain digital elevation models including attributes for surface material and cross country mobility derived from stereo and multi-spectral imagery.

A related program objective is the integration of traditionally independent analytical models, wargames, manned and unmanned simulators, and actual equipment[2] into "seamless simulations" of joint forces in the modern battlefield. Advanced Simulation faces many tough, interesting problems and has joint services interest.

The Special Operations Forces, Aircrew Training System (SOF-ATS) is a large United States Special Operations Command (USSOCOM) program being developed by the Air Force. SOF-ATS will support crew training for the Air Force component of USSOCOM and mission rehearsal for both the Air Force and Army aviation components of USSOCOM, to include numerous Special Operations-variant helicopters and fixed-wing aircraft. SOF-ATS is a broad-based program ranging from specialized computer image generation hardware (CIG) with requirements for very rapid (48 hours) and large (500,000 nm^2) spatial database generation and maintenance. The spatial database will support the simulation of visual, radar, forward looking infra-red (FLIR), and night vision devices to USSOCOM aircrews. SOF-ATS includes the requirement to consolidate intelligence, cartography, imagery, and meteorology source data into an environmental model used by its networked mission rehearsal devices.

As with Advanced Simulation, SOF-ATS raises significant technology issues in the development and maintenance of multi-resolution databases with large areas of coverage. Level-of-detail range from specific buildings modeled at high resolution to coarse resolution terrain simulation over areas covering many thousands of square nautical miles.

There are many additional customers for spin-offs from a program focused on the timely production of cartographic data. These include the DARPA RADIUS program whose focus is on the automated generation and maintenance of detailed site models, the DARPA Unmanned Ground Vehicle program which has several scenarios based upon near real-time update of cartographic databases to support vehicle planning and navigation, and enhancements to the new DMA Digital Production System for improved generation of digital cartographic data at the baseline plant.

Given the current state of the art it can be anticipated that initial operational capabilities for spatial database generation will be largely interactive and labor intensive. However, such systems provide the framework for insertion of emerging technology. A successful program based upon automation will have a profound impact on productivity, flexibility, and timeliness of cartographic compilation in support of a variety of end-users.

3.2. Time Plan and Budget

In order for a new initiative of this type to be successful, it will require a strong DARPA liaison between the image understanding community and key Defense applications. It will be important for the research community to become quite familiar with specific programmatic shortfalls in cartographic data at a level that may require a serious investment of time and resources. Some initial 'windfalls' can be expected from the current image understanding technical base, but a new program will require a significant level of funding ($5-10 million) over two-three years to accelerate the rate of progress so as to impact long-term programs such as Advanced Simulation and SOF-ATS. A portion of this funding can be viewed as 'in place' based upon the DARPA programs in basic IU, UGV, and RADIUS, but these programs have their own needs and goals and can not be relied upon to address these more general issues.

4. Acknowledgements

Many of the ideas in this paper were first discussed at a DARPA IU Principal Investigator meeting held in Santa Fe, NM in April of 1991. Members of the working group included Tom Binford, Connie Gray, Eric Grimson, Tom Henderson, Tod Levitt, George Lukes, Dave McKeown (Chair), Joe Mundy, and Mike Sieverding. Mike Sieverding provided comments on an earlier version of this paper. The authors are solely responsible for the content of this paper and for the emphasis of various topics.

[2] i.e., Army tanks at the National Training Center, Air Force fighter aircraft at Red Flag, and Navy fighters at "Strike University"

Status and Current Research in the Image Understanding Architecture Effort

Charles Weems, Martin Herbordt, Michael Scudder
Computer Science Department
University of Massachusetts
Amherst, MA 01003

James Burrill, Richard Lerner, Thomas Williams
Amerinex Artificial Intelligence Inc.
274 N. Pleasant St.
Amherst, MA 01002

E-Mail Contact: Weems@CS.UMass.EDU

Abstract

The Image Understanding Architecture (IUA) effort is now entering a second phase. The IUA proof-of-concept prototype has been completed and our experience with both the hardware and extensive software simulations are guiding the development of a second generation of the IUA. Furthermore, the initial research-oriented software development environment is currently being replaced by a sophisticated set of application-oriented tools. Thus, the IUA effort is in the process of making the transition from an isolated research project to being in a position of accessibility to the wider community. This article describes the current status of the effort and some of our plans for the future. IUA development is taking place at three sites: the University of Massachusetts at Amherst, Hughes Research Laboratories in Malibu, and Amerinex Artificial Intelligence Inc. The article is thus divided into major sections that describe the efforts taking place at each site.

1. University of Massachusetts

Efforts at the University of Massachusetts have focussed principally in three areas: the design of the second generation IUA hardware, development of advanced programming tools, and algorithm development. The second generation IUA design is nearly complete and, although we expect a few aspects to change, our current view of the architecture is briefly described below. In the area of programming tools, we will give an overview of the multi-associative programming model that we have developed for the low (CAAPP) level of the IUA. We will also discuss some of the issues involved in building a parallel, intermediate-level symbolic representation (ISR) database for the ICAP level of the IUA. We will also summarize an IUA application; that of deriving dense depth maps from known monocular motion.

1.1 Second Generation IUA

For the purpose of comparison, we first summarize the characteristics of the original IUA. The first generation of the IUA is a proof-of-concept prototype containing 4K low-level (CAAPP) processors, 64 intermediate-level (ICAP) processors, and a single high-level (SPA) processor which also serves as the system host [Weems, 1989]. The CAAPP processors are bit-serial, each with 320 bits of on-chip memory and 32K bits of backing-store memory. The ICAP processors are 16-bit TMS320C25 chips, each with 256K bytes of private memory, 256K bytes shared with 64 CAAPP processors, and 128K bytes shared with the SPA. The ICAP processors communicate via a centrally-controlled bit-serial crossbar switch, using their built-in serial communication channels. The SPA is any VME-bus compatible processor, typically a Sun-4.

The Array Control Unit (ACU) for the prototype is a very simple memory buffer for streaming instructions to the array at a high rate. The ACU has no processing or branching capability, and thus all control flow is managed by the host. This arrangement is adequate for its purpose, testing and limited demonstrations of the system, but is not effective for real applications.

Data is loaded into the CAAPP by writing to an image buffer, which is then shifted into all of the processor chips in parallel, via their nearest neighbor mesh. Output from the CAAPP follows the reverse of this process. I/O with the ICAP is performed by the SPA/host, via the dual-ported memory between the ICAP and the SPA/host.

Physically, the IUA prototype consists of 16 12U circuit boards, plus additional boards for control, I/O and communication. The boards are sparsely populated to permit easy diagnosis and rework.

The second generation IUA retains the basic three-level structure of the prototype, but the SPA and host will be separate processors. We expect to use a commercially available multiprocessor board set for the SPA. The host will again interface via a VME extender. The new ACU will be a full-fledged processor, consisting of a microcode engine with a 128-bit instruction word and two separate arithmetic units (one for computation, and one for address arithmetic).

The standard configuration for the second generation IUA will contain 16K low-level processors, 64 intermediate-level processors, and four high-level processors. Physically, the processor array will consist of only 8 12U processor boards, plus some additional boards for control and I/O.

At the CAAPP level, the basic bit-serial architecture will again be used, but a 32-bit corner-turning register increases the on-chip memory to 352 bits per processor. The corner-turning register provides greater flexibility in formatting values that are to be passed to the ICAP. Image I/O with the CAAPP still involves writing to a frame buffer, but the data path to the buffer is now 128 bits wide, permitting a data rate of 160 MB per second. Once the data is in the frame buffer, it appears as merely another segment (HCSM) of backing-store memory (CISM) to the CAAPP processors. Thus, the time to load or store an image is the same as for any other backing store fetch. HCSM provides 4K bytes of storage for each CAAPP processor. CISM has futher been doubled in size to 64K bits (8K bytes) per processor.

The ICAP level has been completely redesigned. It now uses the TMS320C40 32-bit processor, which contains both integer and floating-point units, and operates at up to 50 MFLOPS. Each ICAP processor will have 1 MB of private storage in addition to the ability to access the 2MB of memory it shares with 256 CAAPP processors. ICAP processors are now arranged in groups of four to form a *quadnode* (see Figure 1). Each quadnode has a 4 MB local shared memory which is immediately accessible to the four processors. The local shared memories of all of the quadnodes combine, however, to form a distributed shared memory. Any processor has access to all of the shared memory, although the latency to access a memory outside of the local quadnode will be slightly greater than a local access. In the standard IUA configuration then, there is a 64 MB global shared memory, accessible to all processors. Access to remote segments of the shared memory is via a four by four mesh of buses.

Communication in the ICAP also takes place via a set of message-passing channels. Each processor has six 8-bit channels together with six DMA controllers. Thus, each quadnode has a pool of 24 channels. Of these, 8 form a token ring within the quadnode, 15 are connected directly to all of the other quadnodes, and the remaining channel is

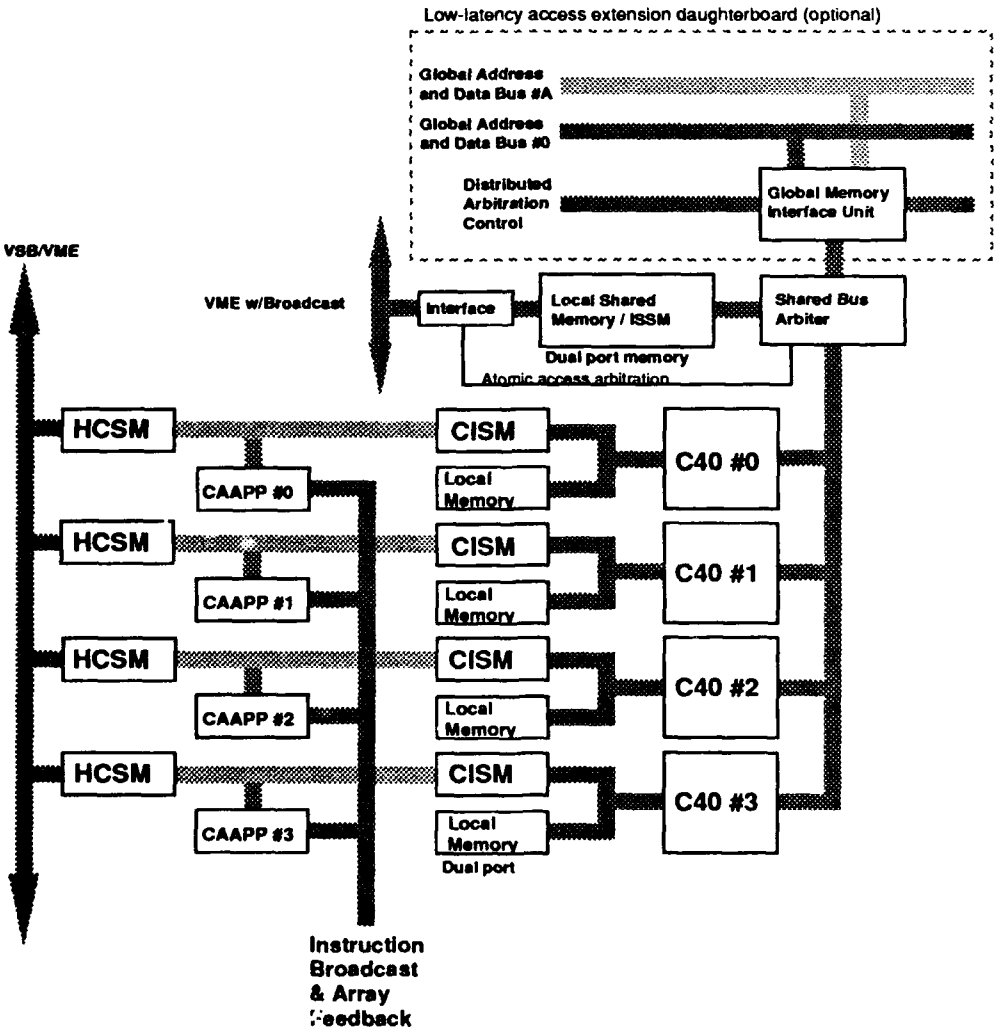


Figure 1. ICAP Bus Structure

brought to an external channel for diagnostics, or custom I/O. Thus, each quadnode is directly connected to all other quadnodes via a DMA channel, as shown in Figure 2.

Note that in Figure 1, a portion of the architecture is labelled as optional. This portion of the system can be omitted in the initial release of the hardware, and then be added later by replacing a daughterboard. In essence, we have designated a minimal subset of the system to reduce the risks in meeting the accelerated development schedule for the second generation. If the optional components are omitted, it is possible to build the entire second generation IUA with

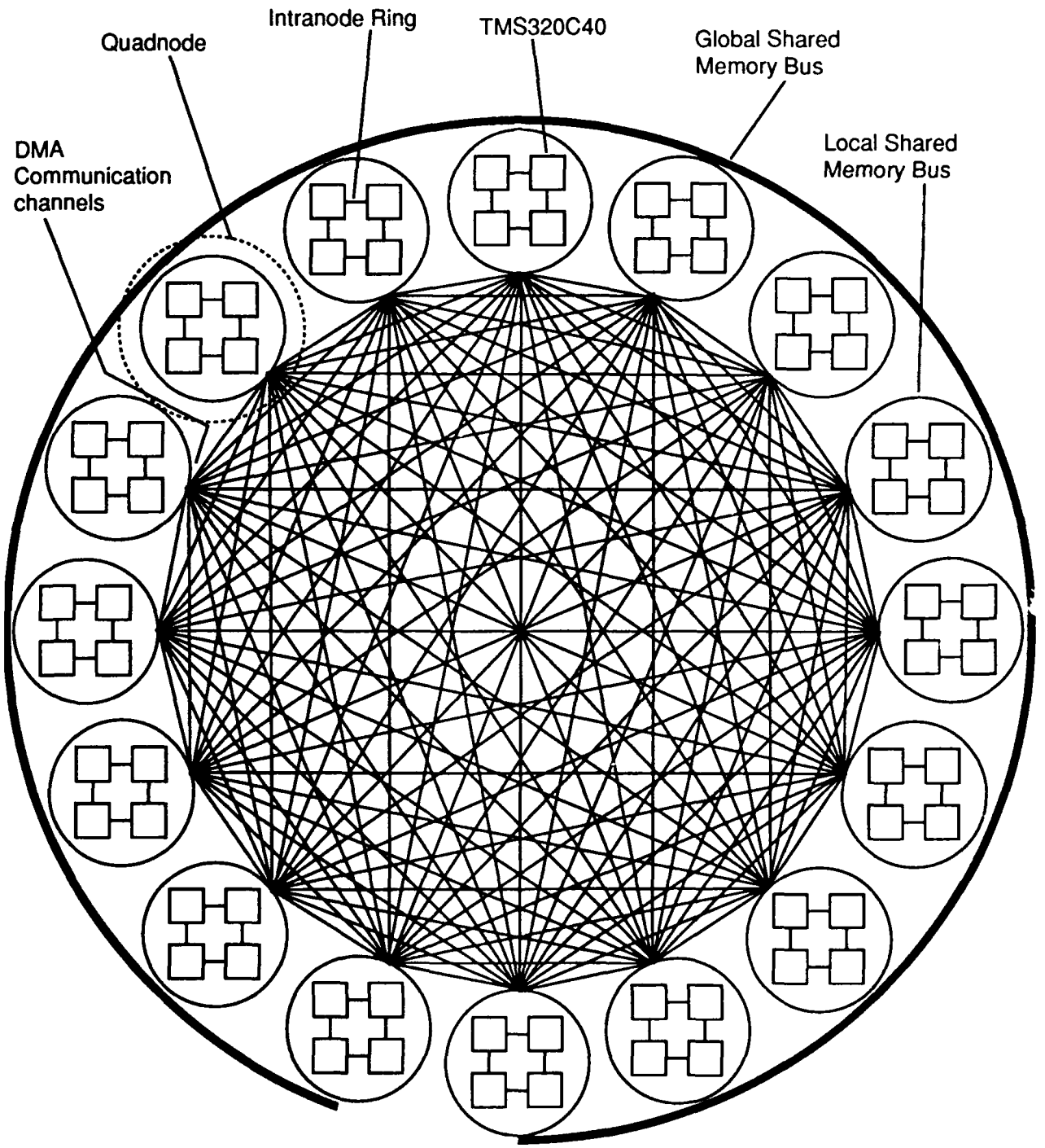


Figure 2. ICAP Communication Structure

off-the-shelf components (except for the custom CAAPP processor chip -- however, that chip is merely a four-times replication of the original CAAPP chip on a single die, so it is also a low-risk component).

1.2 Multi-associative Processing Model

Computation on and among data sets mapped to irregular, non-uniform, aggregates of processing elements (PEs) is an important problem in parallel vision processing, arising in segmentation and in support operations for intermediate-level grouping tasks. The difficulty is that the SIMD processors which map so effectively onto pixel-based processing are restricted in these data dependent computations by the inherent limitations of their control mechanism. Previously, we have used associative processing as a means of applying parallel processing to non-uniform computations [Weems, 1984]. For example, this approach uses global feedback to process individual regions efficiently, but often requires processing to take place on one region at a time. In our current work, we address this problem by introducing an additional level of parallelism, which we call *multi-associativity*, that provides a framework for performing associative computation on independent data sets simultaneously.

A typical vision processing problem to which multi-associativity can be applied is the characterization of regions obtained from a connected components algorithm. Some parameters to be derived may include the number of pixels, boundary length, and mean and median of various spectral quantities. However, since these regions are arbitrarily shaped collections of contiguous processing elements, the communication patterns are also necessarily non-uniform. Although we have developed routing algorithms to collect data using roughly 2d communication steps (where d is the extent of the largest region) [Herbordt, 1990], we would like to take advantage of the constraints provided by this problem to improve that performance.

We first look at how the problem would be solved using traditional associative processing. A typical associative operation is for the controller to broadcast a query to the array, and to receive a response in the form of a count of the PEs with agreeing tag bits. But associative processing, as opposed to the familiar associative memory operations, also enables the conditional generation of symbolic tags based on the values of data, and the use of those tags to constrain further processing. Associative algorithms requiring a number of steps proportional to the number of tag bits have been developed for finding the maximum or minimum value, the mean or median of selected values, and others. Descriptions of these and more complex algorithms can be found in [Foster, 1976; Weems, 1984; and Weems, 1989].

Hardware support in the CAAPP for the global count operation yields performance of approximately 2 micro-seconds; since tag fields are typically 16-20 bits, these associative algorithms complete in roughly 120-200 micro-seconds. Although associative processing enables computations based on PE attributes and relationships to other PEs and events, we are often only processing one region at a time with this approach.

We have developed algorithms for the coterie network [Weems, 1989] to simulate efficiently *within non-uniform aggregates of PEs simultaneously* the associative operations supported directly in hardware by the CAAPP for the entire processor array. Most significantly, we have developed an algorithm to count selected pixels simultaneously in each region in a number of steps proportional to the length of the PE ID (O(logN)). Although this response is not in the micro-second range of the global count, it is significantly faster than previous O(d) algorithms. The consequence is that all existing associative algorithms that were previously run in parallel but region-serially, can now be run region-parallel (on each region simultaneously.) For example, the algorithm to find the mean of some attribute in each region takes O((logN)**2) steps. Although the elapsed time for a single region is significantly longer than the same globally associative algorithm, the gain can still be substantial as often thousands of regions must be processed. We estimate the break even point at between 50 and 100 regions.

Other results we have obtained are new multi-associative algorithms for parallel prefix and convex hull, that is algorithms that perform these computations on aggregates of PEs in parallel and simultaneously. The multi-associative framework also *extends* the traditional associative paradigm by allowing operations on and among aggregates of PEs that are not defined when processing is always performed globally. Two consequences are: the support of divide-and-conquer algorithms within aggregates, and communication among aggregates. The latter operation is especially useful during the merge phase of segmentation algorithms, when characteristics of a region can be transferred to neighboring regions in a single communication step.

1.3 The Object-Oriented Store

A key component of the IUA programming environment is the intermediate-level symbolic (ISR) database [Brolio, 1989; Draper 1989]. The purpose of this database is to store the symbolic representations of extracted image events, groups of events, and instantiated models. As the basis for this new version of the ISR, we are developing a persistent, parallel, object-oriented store.

The store will be object-oriented by Wegner's definition [Wegner 1990], in that it supports objects with classes and inheritance. Objects are the encapsulation of data with the procedures or methods that operate on them. Classes group together objects with a common template. Inheritance uses overloading of functions or operators in a hierarchy of classes to express similarity among related classes.

The store's persistence is derived from the fact that objects may outlive the process that creates them. Thus, programmers are not required to translate between flat file-system storage and structured, encapsulated, in-memory storage.

The parallelism stems from a novel style of parallel programming, known as *POATS*, for Parallel Operators Applied To Structures. POATS provides the speed-up associated with parallel execution but with less programmer effort than traditional MIMD methods. It allows the programmer to concentrate on the data's structure and operations on it, instead of on the coordination of multiple processors. The programmer uses a comprehensive set of data structures and operators to specify transformations of data. POATS is a similar but higher level approach to that of the concurrent aggregates proposed in [Chien 1991].

The operators in POATS can apply functions to data structures in parallel. These functions can themselves be composed of parallel operators so that nested data structures can be handled. Predefined or programmer-created patterns are used to specify dependencies among elements of a data structure. Extensions to the compiler and run-time system of the host language(s) determine the mapping of existing processors to the data structures, and how to coordinate the processors.

POATS combines elements of data parallelism with MIMD processing to permit more flexibility in the manipulation of complex data structures. The POATS model may be thought of as a form of Single Program Multiple Data (SPMD) parallelism, except that it does not necessarily enforce synchronization at the frequent intervals that are typical of SPMD programs. For example, several POATS operations might be active at once, allowing greater utilization of processing resources.

The object-oriented store will eventually become the core of the parallel ISR database. In addition to the capabilities of the object-oriented store, the database will provide meta-data descriptions (schemas), indexing structures, a query language, version control, garbage collection, recovery, and perhaps protection. Built-in objects will include images and image sequences, the DARPA Image Understanding Environment set of standard objects, and libraries of reusable procedures.

1.4 Dense Depth Map Application

An SIMD depth from motion algorithm has been implemented for the IUA using the simulator for the IUA prototype. Image correspondences are established through correlation for two temporally separated images. The depth map is computed from the image displacements and approximately known motion parameters. The map is then filtered to eliminate some possibly erroneous isolated depth values.

The algorithm takes roughly 0.53 seconds to compute on the IUA. By comparison, a similar algorithm for correspondence alone takes about 3 minutes on the Connection Machine, about 10 minutes on a Sequent Symmetry multiprocessor (12 Intel 80386 processors), and about 2 hours on a Vaxstation 3100. The majority of the time is spent on the correspondence, which involves searching a 41 x 41 window in the second image of the pair.

Qualitatively, the algorithm appears to give good results, clearly distinguishing the depths of strong features. Quantitatively, the results are accurate to a range of about 50 feet, for a four foot forward motion of the camera (which has a 45 degree field of view), with a 7.7% mean error in the calculated depths.

2. Amerinex Artificial Intelligence

Efforts at Amerinex Artificial Intelligence (AAI) have concentrated in two areas: completing the design and development of a C++ class library for the CAAPP (initially begun at UMass), and building IU-specific application development tools for the intermediate (ICAP) level of the IUA. Unlike the research-oriented software produced by the University, the effort at AAI is intended to create production-quality software support for the IUA.

2.1 IUA Software Philosophy

Here we describe the progress in creating the software environment that will exist on the IUA. The usual goals were placed on the ensuing environment:

- It should be easy to use.

  As an example, the processing elements (PEs) at the CAAPP level are bit-serial devices requiring 17 operations to add two eight-bit values. It should not be necessary for the user to write these 17 operations or to think of the CAAPP as a bit-serial device.

- It should integrate the various levels of the IUA.

  The IUA consists of several levels of processors and multiple communication paths between and within these levels. It should be possible for users to easily integrate these levels in their problem solutions.

- It should be efficient.

  The incentive to use the combined hardware/software environment, due to increased performance and programmer productivity, should be significantly greater than the incentive to use a standard uniprocessor environment.

- The environment should be familiar.

The user should not be required to learn new and esoteric languages. It is difficult enough to utilize the concepts of parallel processing without having to learn entirely new syntax.

As a general philosophy, we have decided to base the software on objects and C++ where ever this is feasible. As a start, and a framework upon which to tie together the requirements, we developed a Class Library for the IUA. We therefore utilize C++ and a set of classes to describe operations performed at the CAAPP level of the IUA. The programs written using the Class Library look like conventional C++ programs, but where expressions such as (a + b) / (c + 5) may refer to data parallel operations on a SIMD array. These user programs actually run on the Array Control Unit (ACU) and communicate with the host processor using messages (requests). The same programs communicate with the ICAP by invoking processes at that level. We intend to provide libraries that implement various communication schemes. Eventually, to obtain greater object code optimization, we will modify the C++ compiler to recognize our classes, but these changes will not affect the language definition.

In the following section we briefly describe the Class Library for the IUA and how programs written with it communicate with the other parts of the IUA. We describe the major process that runs on the ICAP, and explain how other ICAP processes, both predefined and user defined, are created and communicate.

2.2 The C++ Class Library for the IUA

By using C++, we avoid defining a new language and having to validate its syntax and semantics. C++ provides proven mechanisms for programming that can be used to control the additional operations needed for the IUA. We provide these additional operations using classes (or objects) which are defined using standard C++ and object oriented concepts.

The base class is the *plane*, which is understood to be a two-dimensional grid of elements where each element of the grid exists at a single (virtual) processing element (PE) of the CAAPP. We do not use the term array as it already has a defined meaning in C++ for another construct. In contrast to planes, the nominal objects in C++ are referred to as scalars. Standard arithmetic operations may be applied to planes by applying the operation to each element of the grid that comprises the plane. Standard arithmetic operations may be applied between scalars and planes as well by replicating the scalar for each element of the grid.

Just as scalars are distinguished as being of type *int*, *short*, *char*, etc., planes are also distinguished as being of type *IntPlane*, *ShortPlane*, *CharPlane*, etc. These new classes are derived from the plane class and differ in the number of bits used to represent each element in the grid.

Two levels of control must be provided. The first is "should an operation be applied to an entire plane" and the second is "should an operation be applied to a particular element of a plane". The first type of control is provided by the C++ control statements such as *if* and *for*.

The second method of control is provided by the concept of activity. The activity is specified independently for each virtual PE. Activity is embodied in three new classes *Select*, *SelectNot*, and *Everywhere* which allow the activity to be set for each PE using a BitPlane. Activity controls data transfer within the PE and no other operations. That is, an element in a destination plane is modified under an assignment operation if and only if its PE is *active*. A particular activation has a scope in the same way that variables have scope. Activity may be nested, allowing a cumulative winnowing of the set of active elements.

Other operations are provided for planes. These operations are applied as C++ methods to particular planes. Examples include

- *Any*, which returns a scalar 1 if any elements of a BitPlane are 1 and a 0 if no elements of a BitPlane are 1.
- *Count*, which returns a count of the number of elements of a BitPlane which have the value 1.
- *West*, *North*, *East*, and *South* which implement neighbor communication on the grid.
- Generalized routing operations with combining.

In addition to the the above operations that exist on many SIMD mesh parallel processors, the IUA has hardware for allowing operations to be performed in parallel by regions. This hardware embodies what we believe are important capabilities for image understanding applications. There are four switches at every PE that allow four-way connectedness. A PE is connected to its neighbor if its switch, in that direction, is set and its neighbor's switch, in the opposite direction, is set. Once the switches are set, all PEs that are connected define a region called a *coterie*. Information may be broadcast by some PEs on the circuit formed by the switches and sampled by every PE also on the circuit. Thus, if only one PE per coterie places information on the circuit, it can do a one-to-many broadcast of this information to all the other PEs that form the coterie. If more than one PE broadcasts, the message is the logical OR of the multiple messages sent. For a one-bit message, the result is thus equivalent to the Any operation being applied in parallel to all coteries. Coteries are implemented by the classes *CoterieWENS*, *CoterieWE*, and *CoterieNS* as well as by methods applied to planes.

Because the IUA will exist in several geometries that result in different grid sizes for the CAAPP, it must be possible to write the programs based on the plane size and not the IUA

size. It must also be possible to run the same size problem on both large and small instantiations of the IUA with the only difference being the length of time needed for the computation. A single program may contain several plane sizes. Therefore, programs written using the Class Library specify the size of each plane, and the IUA software maps this to the actual machine that is being used.

For example, if an IUA has a 128 x 128 grid of physical PEs at the CAAPP level and the size of a plane is 256 x 256, then the plane must be split into 2 x 2 tiles to fit the actual IUA. A plane size of 256 x 258 would require 2 x 3 tiles and the tiling factor would be 6.

The size of the plane is specified at compile time by automatically converting the specified size to an integral multiple of the size of the IUA. For example, if the plane size is 40 x 40 and the IUA size is 32 x 32 PEs, there will be four tiles and the actual size will be 64 x 64. We make a distinction between the problem and actual sizes. The programmer must consider the actual size for mesh communications with the *West()*, *East()*, *North()*, and *South()* methods using the torus connections of the mesh. Note that the actual size is the size of the virtual processor array and is not necessarily the same as the physical size.

One of the benefits of this class library is that it does not require the use of an IUA. The class library can be implemented on other SIMD architectures with more or less ability to support the operations provided by the library. It has also been implemented on sequential machines. The generality of the Class Library for the IUA allows it to form the basis of a language for specifying a wide range of image understanding algorithms.

Figure 3 is an example function which calculates the integer square root for each element of an IntPlane. Note how similar this function is to one for scalars.

The function shown in Figure 4 implements a simple edge operation in the x-direction, and is an example of neighborhood communication.

The function in Figure 5 forms regions based on connected component equivalence classes and then labels the regions formed using the address of one of the PEs in each region; it is an example illustrating the use of coteries.

2.3 The ICAP and the ISR

Software for the IUA's intermediate (ICAP) level is arranged hierarchically with each layer providing additional functionality or an abstraction of the lower levels.

Figure 6 depicts the hierarchy. This section describes our current designs for each layer. None of the components have been implemented yet.

```
ShortPlane
IntSqrt(IntPlane initial)
{IntPlane guess (initial.Size());
IntPlane last_guess(initial.Size());
IntPlane res     (initial.Size());
BitPlane a       (initial.Size());
int iterations = 18;
int count;

  guess = initial;
  a = (guess != 0);
  count = a.Count();
  {Select active(a);
   while (iterations--) {
     last_guess = guess;
     {Everywhere active;              // Set 0s to 1s
      {Select active(guess == 0);     // because divide will
        guess = 1;                    // be done everywhere
       }
      }
     res = initial / guess;
     res += last_guess;
     guess = res >> 1;
     if (count <= (guess == last_guess).Count()) break;
     }
   }
  return ShortPlane(guess);
}
```

Figure 3. C++ Class Library Example of Calculating the Integer Square Root of Every Pixel in an Image

```
ShortPlane
prewitt_x(UCharPlane image)
 {ShortPlane x(image.Size());
 //  Compute the first derivative in the X axis direction
 //  with a simple edge operator that applies this mask:
 //   -1 -1 -1
 //    0  0  0
 //    1  1  1
   x = image.South() - image.North();
   return x + x.West() + x.East();
}
```

Figure 4. C++ Class Library Example of Neighbor Communication

A user's ICAP program consists of a set of entry points; the ACU causes the intermediate level to begin execution at one of these entry points as part of the execution of the user's ACU program.

Once begun, the ICAP program performs some complex operation which may involve communication with other portions of the IUA, communication among ICAP processors, maintenance of a shared database, servicing interrupts, and starting or interacting with additional threads of control (tasks). The program performs these actions with the help of the software components in the hierarchy.

```
// Segment `equivalence classes' into regions by
// comparing the values of neighboring
// PEs and then label each region.

IntPlane
label_regions(UCharPlane eq_class)
 {Everywhere active;     // Insure that every PE participates.
  BitPlane west   (eq_class.Size());
  BitPlane east   (eq_class.Size());
  BitPlane north  (eq_class.Size());
  BitPlane south  (eq_class.Size());
  BitPlane masters(eq_class.Size());
  IntPlane labels  (eq_class.Size());

  // Determine the switch settings for the coteries
  // Do not wrap regions around the grid edge

  west  = (eq_class == eq_class.West ()) &
          ~eq_class.WestEdge_p();
  north = (eq_class == eq_class.North()) &
          ~eq_class.NorthEdge_p();
  east  = (eq_class == eq_class.East ()) &
          ~eq_class.EastEdge_p();
  south = (eq_class == eq_class.South()) &
          ~eq_class.SouthEdge_p();

  // form the regions
  {CoterieWENS pattern(west,east,north,south);

  // Select the active PE with the highest
  // address in a region.
  masters = (eq_class.Index()).RegionSelectMax();

  // Label each PE with the address of the master PE.
  labels = (eq_class.Index()).RegionBroadcast(masters);
  }
 return labels;
}
```

Figure 5. C++ Class Library Example of a Coterie Operation (Connected Component Labelling)

At the bottom of the hierarchy lies an ICAP processing element. These processing elements are arranged into groups of four (quadnodes) with each quadnode linked to every other quadnode via the communication ports of the processing elements (and by a shared memory structure).

On top of the processing elements we provide basic system runtime support using SPOX, a real-time multi-tasking operating system sold by Spectron Microsystems, Santa Barbara, CA. SPOX is a widely used commercial product for real-time applications on digital signal processors. It is expressly designed to be easily ported to architectures such as the IUA. SPOX provides basic system support functions such as simple preemptive scheduling, software interrupts, efficient I/O, management of multiple memory segments, and other functions, with low overhead. SPOX provides the basic tools with which we build the higher-level abstractions that are appropriate to our programming domain, including custom multi-tasking abstractions, and synchronization and communication constructs.

The Task Management layer is an interface to many of the SPOX features concerning tasks. This layer provides both basic routines to start tasks, change priorities, and check on the status of tasks, and more abstract routines such as those for executing functions as separate tasks (i.e., "background processes"). In addition, this layer provides a framework within which interrupts are defined and attached to events, and the framework in which a user's program defines its entry points. Finally, it defines such abstract objects as mailboxes, monitors, and other objects to manage synchronization.

The Communication layer defines the basic routines for communicating with other parts of the IUA. For communication via the processor ports, it presents a simple message passing interface; there are routines to construct messages, have them sent to one or more destinations, and

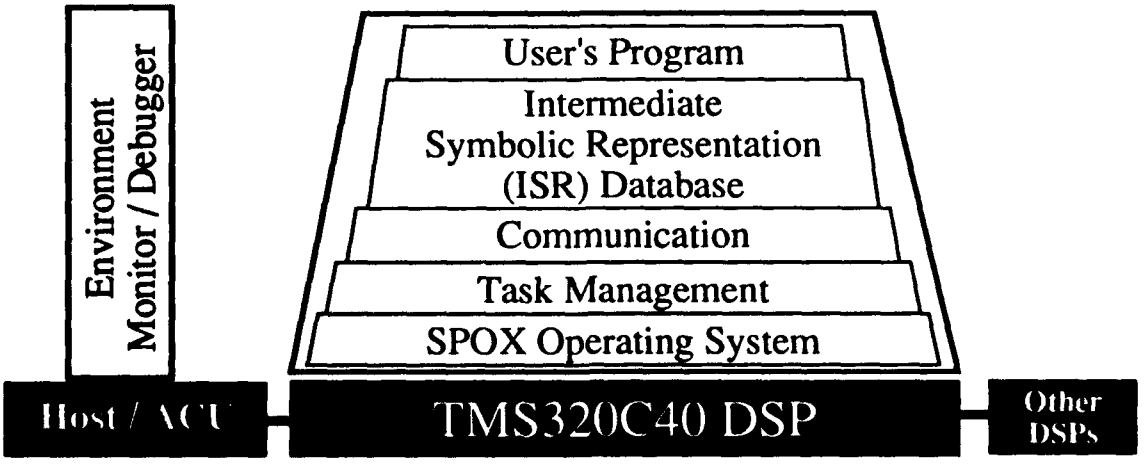


Figure 6. ICAP Software Hierarchy

to receive and dispatch them to the appropriate message handler. For communication via the shared memories, this layer presents both a message passing interface where certain memory locations contain the message queues, and block read/write functions. Both interfaces provide operations that also hide much of the complexity of ICAP--CAAPP communication when running programs with plane sizes larger than the physical CAAPP array. The communication layer also defines such abstract objects as global shared variables.

Although the previous layers are sufficient for intermediate level programming, the interfaces they provide are still fairly primitive. The Intermediate Symbolic Representation (ISR) Database layer provides a higher-level framework within which the ICAP processes can work: that of a shared database of tokens representing image components at various levels of granularity. The usefulness of the ISR database as a framework for image understanding programs has been demonstrated through its use at the University of Massachusetts, where reasearch versions being developed, and as an integral part of KBVision, a commercial research tool for image understanding programming that is a product of Amerinex Artificial Intelligence. The ISR database implemented in this layer is based on that of KBVision. The most significant extensions are those to support a distributed database with limited memory, issues that were not of concern when developing KBVision's ISR. Section 2.3.1 describes the ISR and our extensions.

The final software component in Figure 6, the Programming Environment, provides the tools necessary to run and debug ICAP programs. These tools run on the host machine and interact with the ACU and shared memories. They provide support for loading ICAP programs onto the ICAP processors, loading data into both the shared and local memories, executing initialization routines, and saving memory data after the program has run. Despite the support of the other layers, we expect ICAP programs to be difficult to debug; we therefore require good debugging tools. These tools consist of a portion that runs on the host or ACU and another portion that runs on the ICAP processors. Together they support ICAP program I/O and possibly an interactive debugger. When available, the debugger would allow the setting of breakpoints, program single stepping, and examining memory and registers. The debugger must also be able to handle debugging programs running on multiple ICAP processors simultaneously.

Our layering of the ICAP level software provides a familiar programmer's model (the ISR database) for doing typical processing while providing support for more complex but possibly more efficient management of and communication among the processes running at the ICAP level.

2.3.1 ISR Database

The ISR Database is a repository for information representing abstract image events, such as lines, regions, and edges. It provides tools for defining, storing, retrieving, manipulating, filtering and organizing these events using an interface derived from the ISR database in KBVision [Amerinex, 1991]. Within the database a token describes an image event with a set of features and its spatial location. A tokenset is a group of similar tokens, such as the tokens representing lines extracted by a particular algorithm. A common operation is to find the tokens in a tokenset whose feature values satisfy some criteria. The results of such an operation is a tokensubset, a possibly empty subset of the tokens in the initial tokenset.

On the IUA, a tokenset may be distributed over multiple quadnode IPSMs (memory shared among the ICAP processors in a single quadnode). In a typical operation, ICAP processing begins by populating tokensets with tokens. Since all ICAP processors generate tokens in parallel based on their portion of the image data, the tokens naturally distribute among the quadnodes. This distribution provides a natural partition for parallel computation, but complicates global processing on the tokenset. Our interface supports use of the natural parallelism and hides many of the details of global access.

After populating tokensets, each ICAP typically creates its own tokensubsets for analysis. This operation requires communication among the quadnodes in order to gather token information from the other quadnodes. An ICAP broadcasts a request to every quadnode and constructs the tokensubset with the token information that is returned. Since tokensubsets are unordered, an ICAP can begin processing a tokensubset as soon as data arrives, suspending when it reaches the end of the available data. In the general form of this operation, an ICAP broadcasts to the quadnodes an arbitrary, although previously defined, function and receives a list of the results, which may or may not be tokens. The first form gathers data for processing while the second distributes processing and gathers the results. The general form of the operation may be more efficient than the first depending both on the relative sizes of the input data and the results, and on the data needs of future operations. A process that applies a function to a small number of tokens or that performs multiple operations on a set of tokens may want to gather the tokensubset locally and perform the operations. Broadcasting the function may be more efficient for a process that applies only one function to a large number of tokens, returning a small result (e.g., a summation). We provide an interface that hides many of the details of the tokensubset-specific form. This interface is based on the DynamicList object that implements the more generic form.

2.3.2 Tokensubset Queries

From the user's perspective, tokensubset creation on the IUA is similar to that within KBVision; the user provides a set of tokens and membership criteria and gets a tokensubset in return. However, when using compound criteria, such as those based on multiple token features, the KBVision user creates a tokensubset by successively adding or removing tokens from tokensubsets, while the IUA user creates a

criteria record describing the compound criteria and broadcasts only one request.

Our interface allows tokensubset processing to begin before the entire tokensubset has been received. From the user's perspective, the only change is an additional parameter to all of the tokensubset access functions that specifies whether to block or return when the requested data is not yet available. Non-blocking operations give a "data not available" return code if data is unavailable.

A tokensubset is a simple list of identifiers of tokens in a tokenset. A process uses these identifiers to access the tokens' feature data stored in the tokenset. We provide a form of lazy evaluation of feature values to reduce communication. If a process attempts to access the feature of a token stored in a different quadnode, the system communicates with the remote quadnode to get the information. We then cache this information for future reference. A user's program can prefetch feature data by specifying a list of features when broadcasting its tokensubset criteria. The user's program can also indicate which features are or are not of interest by using some additional directives. Each quadnode records to whom information is sent so that their caches can be updated or invalidated as features change.

The IUA has a limited amount of memory for storing a quadnode's tokens and caching other quadnodes' token data. Long-lived image understanding processing requires some form of garbage collection. Initially, the user's program will be responsible for managing database memory. The user's program running on the ACU is primarily responsible for allocating and freeing tokensets. The ACU allocates tokensets so that they are known to all of the ICAPs. When some phase of processing is complete, the ACU frees the tokensets that are no longer needed. Freeing a tokenset frees the memory associated with the tokens in the tokenset and frees any cached information about tokens in the tokenset. Tokensubsets that are local to an individual ICAP must be freed by that ICAP. Tokensubsets that have been stored as part of a token feature will be deleted when the token's tokenset is deleted. ICAPs also have control over their own quadnode's token data cache through directives indicating when tokens or token features are no longer needed.

Figure 7 shows some of the tokensubset criteria record operations. This and the following examples use C++ syntax although the decision on whether to use C++ or C has not been made yet. When creating a criteria record, the user specifies the tokenset whose tokens are candidates for inclusion within the tokensubset. The user specifies the tokensubset criteria with calls to the *Add* and *Op* operations. The Add operation specifies one criteria and Op specifies the boolean conjunction of multiple criteria. In the arguments to Add, Operation specifies whether matching tokens should be added to or removed from the result, and Test specifies which of a number of tests to use to determine matches. As in KBVision, the test can be one of the following:

- *All*: Match all tokens in the tokenset.
- *Value*: Match tokens whose value in a particular feature is within some bounds.
- *Location*: Match tokens with a location feature that intersects a specified rectangle.
- *Undefined*: Match tokens for which a particular feature value is undefined.
- *NotComputed*: Match tokens for which a particular feature value has not been computed.
- *Criteria*: Match tokens which match a previously defined criteria record.

```
class TssCriteria
 {  ...
  TssCriteria new ( TokenSet );
  void Add ( Operation, Test, TestArgs );
  void Op ( BooleanOp );
  Tokensubset Broadcast ( Destinations, BlockTimeOut );
 };
```

Figure 7. Tokensubset Criteria Operations

The choice of Test determines the arguments that follow. All of the tests, with the exception of All and Criteria, require a feature name and a range of values as the next arguments. All takes no additional arguments and Criteria takes another criteria record as the following argument.

The argument to Op is one of the boolean logic operators: And, Or, or Not. These operate on the "stack" of criteria entered with the Add operation, allowing arbitrary criteria combinations. For example, the following represents a disjunction of two conjunctions:

```
crit.Add(...);
crit.Add(...);
crit.Op(And);
crit.Add(...);
crit.Add(...);
crit.Op(And);
crit.Op(Or);
```

The Broadcast operation sends a completed criteria record to some subset of the quadnodes, and either waits for replies or returns immediately, as specified.

Tokensubset intersection and union are local operations invoked on tokensubsets after they have been received. Thus, these operations cannot be specified in criteria records.

The code in Figure 8 demonstrates the use of tokensubsets and criteria records. This code creates a tokensubset containing the tokens in the tokenset Lines that have start or end points near the start or end points of the line keyLine and have a contrast (indicated as a floating point number) similar to that of keyLine. The first lines of code define the criteria test parameters. The variables dist and conRange

```
ISRFLOAT sminX, smaxX, sminY, smaxY;          \\ area around keyLine's startpoint
ISRFLOAT eminX, emaxX, eminY, emaxY;          \\ area around keyLine's endpoint
ISRFLOAT minCon, maxCon;                      \\ range of acceptable contrast
sminX = keyLine.StartPoint.x - dist;
smaxX = keyLine.StartPoint.x + dist;
sminY = keyLine.StartPoint.y - dist;
smaxY = keyLine.StartPoint.y + dist;
eminX = keyLine.EndPoint.x - dist;
emaxX = keyLine.EndPoint.x + dist;
eminY = keyLine.EndPoint.y - dist;
emaxY = keyLine.EndPoint.y + dist;
minCol = keyLine.Contrast - conRange;
maxCol = keyLine.Contrast + conRange;         \\ Create Tokensubset membership criteria record
TssCriteria crit(Lines);                      \\ criteria record for subset of tokens in Lines
crit.Add(InsertWhen, Location, "StartPoint", sminX, smaxX, sminY, smaxY);
crit.Add(InsertWhen, Location, "StartPoint", eminX, emaxX, eminY, emaxY);
crit.Op (Or);
crit.Add(InsertWhen, Location, "EndPoint", sminX, smaxX, sminY, smaxY);
crit.Op (Or);
crit.Add(InsertWhen, Location, "EndPoint", eminX, emaxX, eminY, emaxY);
crit.Op (Or);
crit.Add(InsertWhen, Value, "Contrast", minCon, maxCon);
crit.Op (And);                                \\ Get Tokensubset
tss = crit.Broadcast(AllQuadNodes, NoBlock);
for(TssIndex = 0; tss.IsIndex(TssIndex, Block); TssIndex )
 {TokenIndex = tss.GetTokenIndex(TssIndex, Block );
  xx = Lines.GetFeature (TokenIndex, "FeatureX", Block );
  < token computation >
 }
```

Figure 8. Example of Tokensubset Usage

define "near" and "similar to". The line beginning with TssCriteria creates a new criteria record, crit, for specifying match criteria for tokens from the tokenset Lines. The first call to Add specifies a criteria that matches tokens in Lines whose starting point is near keyLine's starting point. The following three lines add criteria that match the other combinations of start and end points. The Or operations indicate that a token matches this criteria if either of its points are near either of keyLine's points. The last Add specifies a criteria that matches tokens with similar contrast. The And operation that follows this Add specifies that matching tokens must both be near and of similar contrast.

After completing the criteria record, we create the tokensubset by broadcasting the criteria to all quadnodes (including the local quadnode), returning without waiting for the replies. The for-loop iterates over the tokens in the tokensubset. IsIndex returns true if the index is within the range of the tokensubset. If some replies are pending and the index is beyond the range of the elements present, this operation will block until an element appears for that index position, or all replies have been received and the index is still out of range. The first line in the body of the loop determines the index within the tokenset Lines of the specified element in the tokensubset. The next line uses the tokenset index to get a particular feature. If the feature is not available locally this operation blocks until it is received. Replacing Block with Timeout(time) would allow these operations to return with status "data not available" if the data was not available within the specified timeout period. NoBlock is synonymous with Timeout(0).

2.3.3 Dynamic Lists

Underlying the tokensubset interface are dynamic lists, a general request/reply mechanism. A dynamic list object assists a program with broadcasting a request to other quadnodes and gathering the replies. The program specifies a function to run on the remote ICAP processor, and a handler to append replies onto a list as they arrive. The system broadcasts this request to the other quadnodes and routes the replies to the appropriate handler. After sending the request, the program may begin processing the list of results as soon as data begins to arrive.

Figure 9 demonstrates the use of dynamic list objects from the caller's perspective. This function finds the token with the greatest contrast. Rather than asking for a tokensubset containing all of the tokens and then invoking FindMaximum on the result, this function asks each quadnode to compute the maximum of its local tokens and reply with this token. The result is a list of tokens, one from each quadnode. The function then invokes FindMaximum on this smaller tokensubset and returns the result. This function also asks for the Contrast and Length features of each quadnode's maximum token.

```
TokenIndex FindMaximumContrast ( tokenset )
  Tokenset tokenset;
{DynamicList dynl ();
dynl.Function ( FindMax );
dynl.AddArgs ( String, tokenset.Name );
        \\ TokensetName
dynl.AddArgs ( String, "Contrast" );
        \\ Feature
dynl.AddReplyFeature ( "Contrast" );
        \\ ReplyFeatures
dynl.AddReplyFeature ( "Length" );
dynl.Handler ( TokensubsetHandler );
        \\ Handler for replies
tss = (Tokensubset) dynl.Broadcast (AllQuadNodes,
        NoBlock);
maxEltTss = tss.FindMaximum(Contrast,Block);
maxEltIndex = tss.GetTokenIndex ( maxEltTss, Block );
return (maxEltIndex );
}
```

Figure 9. Example of Dynamic List Usage

The FindMaximumContrast function first creates a new dynamic list object. It then specifies the function to call and its arguments. It specifies the features what should be contained in any token in the reply, and a function to handle replies. Since the replies will contain token information, the function uses the handler for receiving tokensubset data; the result of the handler will be a tokensubset (i.e., a list of token indices). The function then broadcasts the request, computes the maximum of the tokens in the resulting tokensubset, and returns its tokenset index. The call to Broadcast uses NoBlock so that the following call to FindMaximum can begin its processing as soon as data arrives. This call uses Block so that it waits for all the data to arrive before returning its final result. Using a timeout would let the function return the maximum of the data already received, with a return status of "data not available".

Figure 10 shows the FindMax function that runs on the remote quadnodes as a result of the dynamic list request. The first argument contains information needed when replying to the request. The system provides this argument when it calls the function. The remaining two arguments were provided when initializing the dynamic list object. The function gets the indicated tokenset and uses a criteria record to create a tokensubset containing the tokens for which the feature has a defined value. The argument to Broadcast specifies that only local tokens should go into the tokenset. The function then finds the token with the maximum feature value and creates and sends a reply using the token's tokenset index. The AddData operation automatically inserts the Contrast and Length features into the reply message according to the dynamic list request. The argument to Send can be either Complete or Partial. Complete specifies that this is the last reply from this processor for this request. Partial specifies that more messages will follow. Partial replies allow the originating ICAP to begin processing on partial results without waiting for a long process to complete.

Figure 11 shows the handler for this dynamic list object. The system calls this handler for each reply. The handler returns a dynamic list which is given back to it with the next message. The second argument contains the data in the reply (i.e., the result of the AddData operation in Figure 10). The handler calls cachePartialToken to cache the token feature data in the local database, and appends the tokenset index onto the dynamic list.

```
void
FindMax (requestHeader, tokensetName, feature )
  ...
{
  \\ Get the tokenset based on its name.
   tokenset = GetTokenset (tokensetName );
  \\ Create Tokensubset -- Match all
  \\ tokens with feature defined.
   {
    TssCriteria crit(tokenset);
    crit.Add ( InsertUnless, Undefined, feature );
    tss = crit.Broadcast ( Local );
    }
  \\ Get TokenIndex of token with maximum
  \\ value in feature.
  maxEltTss = tss.FindMaximum ( feature, Block );
  maxEltIndex = tss.GetTokenIndex ( maxEltTss, Block );
  \\ Create reply record.
   {
    DynamicListReply reply( requestHeader );
    reply.AddData ( PartialToken, tokenset, maxEltIndex );
    reply.Send ( Complete );
   }
}
```

Figure 10. Example Dynamic List Function -- Return Token With Maximum Feature Value.

```
DynamicList
TokensubsetHandler (dl, msg );
  DynamicList dl;
  DynamicListMsg msg;
{
  while ( msg ) {
   switch ( msg->type ) {

     case PartialToken:
      cachePartialToken ( msg->partialTokenData );
      dl.Append ( msg->partialTokenData.TokensetIndex );
       break;

     default:
      error ( "unknown message data type" );
      }
   msg++;
   }
 return ( dl );
}
```

Figure 11. Example Dynamic List Reply Handler -- Tokensubset Handler

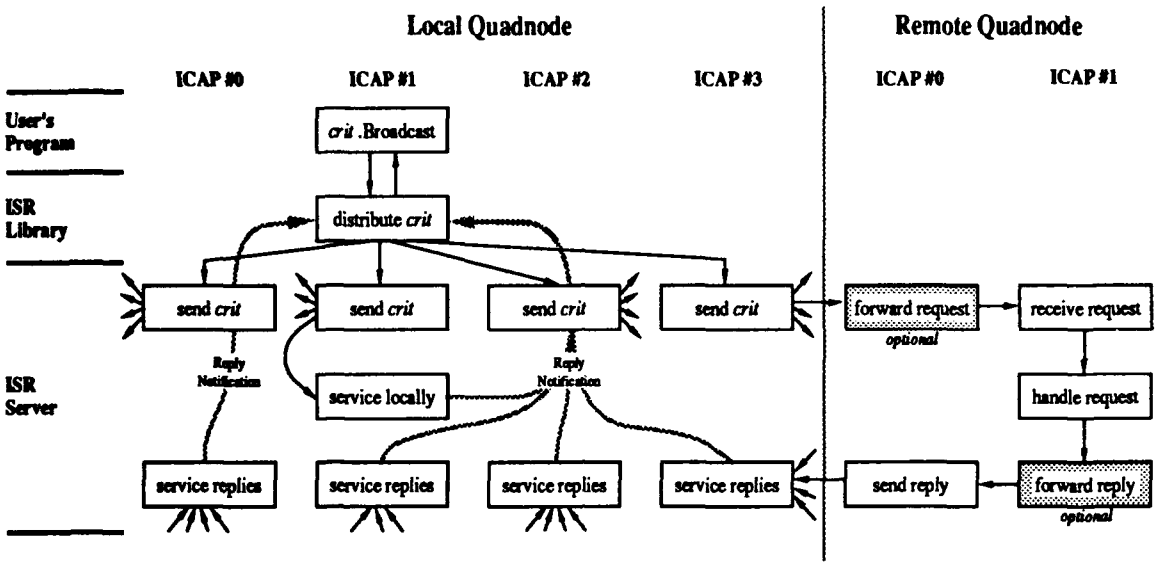


Figure 12. ICAP Communication Example

In this example the resulting dynamic list is a tokensubset that can be accessed with the tokensubset commands discussed above. In other cases, the dynamic list is accessed through its own commands, including indexed access, mapping functions to invoke a function on each element, status functions (e.g., current length and replies pending), and list deletion.

2.3.4 Low-level Communication

Figure 12 depicts the communication resulting from a tokensubset query. The communication lines between quadnodes are distributed among all of the processors in a quadnode. Thus, to send a message to a particular quadnode, the message must first be sent to the processor on the sending quadnode that has the line to the receiving quadnode. Broadcasting messages to all quadnodes requires giving the message to all of the processors on the sending quadnode.

At the top of Figure 12 the user's program invokes the Broadcast operation on a criteria record. This operation is part of the ISR client library and runs as a subroutine call within the user's program. It distributes the criteria record to, and requests service of, each processor on the local quadnode. The ISR server task running on each processor responds to the service request by sending the criteria record to each of its connected quadnodes and preparing to respond to their replies. In addition, one processor is responsible for servicing the criteria request locally as if it had come from a remote processor. As replies arrive, the ISR server task routes the messages to the appropriate handler and notifies any blocked tasks that more data is available. On a remote quadnode, an ISR server task running on the processor with the link to the originating quadnode receives the request and either services it directly or forwards it to another processor in the quadnode. A reply is then sent back to the originating quadnode. If the remote quadnode forwards the request to another processor, that processor must forward the reply to the processor with the link to the originating quadnode, which then sends the reply. The shaded boxes in the figure represent the optional forwarding operations.

All communication among ICAP processors behaves in a manner similar to that of a tokensubset query. Dynamic list requests are nearly identical. For point-to-point communication (from one processor to a particular processor on a remote quadnode) the ISR library routine gives the request to the single processor with the link to the remote quadnode, rather than to all processors; the remainder of the communication is unchanged.

2.4 Inter-level Communication

In this section we describe our plans for other aspects of the software environment being constructed for the IUA. Please note that because the CAAPP is a SIMD array of processing elements whose instructions are generated by the ACU, we treat the ACU and the CAAPP as being one unit in this discussion.

2.4.1 Between the Host and ACU/CAAPP

The Host to IUA connection is that of general purpose computer with a special purpose attached processor. Communication consists of requesting that the IUA, through the ACU, perform some task and return the results of performing that task back to the host. That is, the IUA is an allocated device that performs very complex tasks instantiated in the form of large programs that run on the ACU. Therefore, the host must have a means of initiating tasks on the ACU and then communicating with those tasks as they are running. The host will be executing processes

under some version of Unix. The ACU will be executing tasks under some real-time executive (such as VX Works). Communication will be in the form of messages sent via sockets on the host side and implemented through library functions callable from higher level languages. On the ACU side, this communication will utilize the same semantics but be implemented via a separate library callable from programs written using the Class Library for the IUA.

In a developmental environment, the users will interact with the host and thus with their tasks on the ACU through normal input and print statements. The library for programs running on the ACU will implement the standard C++ stream library functions to provide this interactive facility.

2.4.2 Between the ACU/CAAPP and ICAP

There are three mechanisms for communication between the ACU/CAAPP and the ICAP: via the shared memory layers above and below the ICAP, and via ACU broadcast.

There is a layer of memory, called the CAAPP ICAP Shared Memory (CISM), that resides between the CAAPP and ICAP levels of the IUA and is read/write accessible by both. This memory is used by the ACU/CAAPP for the storage of planes which allows these planes to be available to the ICAP processor and to serve as a means of communication between these levels.

The ACU is capable of broadcasting information to all ICAP processors simultaneously. Programs to be run at the ICAP level are loaded and initiated using this mechanism. Issues of synchronization are handled by library routines available to programs written using the Class Library for the IUA. These routines are based on the broadcast mechanism and the ability to interrupt the ICAP processors.

The ACU also has access to another layer of memory, called the ICAP-SPA Shared Memory (ISSM), accessible by the ICAP and SPA processors. ISSM is addressable from the ACU using a function based upon an ICAP processor's address. It may be used by the ACU to make requests to a particular ICAP processor. The same memory may be used by an ICAP processor to return results or make requests to the ACU (in conjunction with CISM and the CAAPP global feedback mechanism).

2.4.3 Between the ACU/CAAPP and Sensors

The ACU determines when input images are sent to the Host-CAAPP Shared Memory (HCSM) and when images are sent from the HCSM to the outside world. This control is exercised by the user's program through another library that contains routines to control the IDTS (Image Data Transfer System). These routines control the underlying hardware, accessing it via the VME bus. The host also has access to the hardware through the VME bus. But, issues of synchronization with the CAAPP require that only the ACU exercise this control.

2.4.4 Between the Host and ICAP

In normal cases, we do not expect that the host will have a need to directly communicate with the ICAP level of the IUA. While the host has access to some of the same memory that is available to the processors at the ICAP level, issues of synchronization make it unlikely that this facility will be used.

3. Hughes Research Laboratories

Efforts at Hughes Research Laboratories have been directed mostly towards debugging the IUA prototype hardware, designing the ACU for the second generation IUA, the new custom CAAPP chip, and the overall second generation IUA architecture.

The IUA prototype became operational in June of 1991 but, as with most prototype efforts, several problems were encountered. Two of the more serious problems were related to subtle errors in the custom CAAPP chip that were not detected in the circuit simulations. One of the errors involves a control line that passes underneath a portion of the on-chip memory and can cause bits to be lost due to parasitic effects. This error has since been corrected in the second generation CAAPP chip. The second problem involves ground-loop noise due to the spacing of ground pins in the carrier, and will be alleviated by rerouting all ground lines to the inner ring of pins in the second generation chip.

Other problems included resolving interference between Unix and the software initiated memory refresh (refresh is now generated in hardware), compensating for clock skew in the system, and repairing numerous unreliable solder joints.

A preliminary version of the C++ class library, together with the IUA prototype simulator, has been used to develop a missile-tracking related demonstration which has been run successfully on the hardware. In addition, numerous testing and diagnostic routines have been run, and further software is being developed to exhaustively exercise the prototype.

As stated in section 1, a new ACU has been designed that will include a 128-bit horizontal microengine built from AMD 29000 series bit-slice logic. The microengine contains much of the run-time library for the IUA, and is capable of issuing instructions to the CAAPP and ICAP arrays as quickly as they can accept them, and with very little overhead. The instruction issue rate of the ACU is decoupled from the execution rate, and instructions are actually issued asynchronously.

The ACU also contains a "macroengine" consisting of a single-board computer based on a SPARC processor. The macroengine executes the high-level control portion of the user's program and issues instructions to an abstract machine consisting of the microengine and its subroutine library. Thus, a macroengine command might be to perform floating

point division of one plane by another, and the microengine will expand this into the appropriate stream of instructions for the CAAPP array.

Hughes Research Laboratories has also participated in the design of the second generation IUA architecture, developing a separate initial proposal from that of UMass. Ideas from both proposals were combined into the design presented in section 1, and as mentioned there, a few of the details for the design remained to be ironed out, but we expect it to be completed before the end of 1991.

Other efforts at Hughes include the design of the IDTS (which is partially dependent on the specification of the DARPA UGV sensor suite), and advanced packaging for the IUA to further reduce its size while increasing the number of processors.

4. Conclusions

The Image Understanding Architecture is undergoing significant change with the development of the second generation. Both the hardware and the software are being substantially enhanced. We expect the new system to be computationally more powerful than the prototype, and to be much easier to use for vision applications. In particular, the new IUA is being targetted for use in the DARPA UGV program, which will present a wide range of applications challenges, and no doubt lead to further insight into the means for exploiting the potential parallelism in image understanding.

5. Acknowledgements

This work was funded in part by the Defense Advanced Research Projects Agency under contral number DAAL02-91-K-0047, monitored by the U.S. Army Harry Diamond Laboratory, contuact numbers DACA76-86-C-0015, and DACA76-89-C-0016, monitored by the U.S. Army Engineer Topographic Laboratory; and contract number F49620-86-C-0041, monitored by the Air Force Office of Scientific Research; and by a Coordinated Experimental Research grant (DCA 8500322) from the National Science Foundation. The authors thank David B. Shu, John Spalding, Dennis Finn, Howard Neely, and J. Gregory Nash at Hughes Research Laboratories for their contributions to the IUA design and development. Thanks also to Rabi Dutta for developing the parallel dense depth map application.

6. Bibliography

[Amerinex 1991] Amerinex Artificial Intelligence, "The KBVision System, Programmer's Reference Manual", Release 2.4, Amerinex Artificial Intelligence, Amherst, MA, June 1991.

[Brolio, 1989] John Brolio, Bruce Draper, J. Ross Beveridge, Allen Hanson, "ISR: A Database for Symbolic Processing in Computer Vision", IEEE Computer, Vol. 22, No. 12, pp. 22-30, December 1989.

[Chien 1991] Andrew A. Chien, Concurrent Aggregates: Using Multiple-Access Data Abstractions to Manage Complexity in Concurrent Programs, OOPS Messenger Vol 2 No 2, April 1991.

[Draper, 1989] Bruce Draper, et al., "The Schema System," International Journal of Computer Vision, Vol 2, No. 3, pp. 209-250.

[Foster, 1976] Caxton C. Foster, Content Addressable Parallel Processors. Van Nostrand Reinhold Company, New York

[Herbordt, 1990] Martin Herbordt, Charles Weems, Jay Corbett, Message Passing Algorithms for a SIMD Torus with Coteries. Proceedings of the 2nd ACM Symposium on Parallel Algorithms and Architectures, pp 11-20. Also in Computer Architecture News, Vol. 19, No. 1, pp. 69-78.

[Weems, 1984] Charles Weems, Image Processing on a Content Addressable Array Parallel Processor. Technical Report 84-14, Department of Computer and Information Science and Ph.D. Dissertation, University of Massachusetts

[Weems, 1989] Charles Weems, Steven Levitan, Allen Hanson, Edward Riseman, J. Gregory Nash, David Shu, The Image Understanding Architecture. International Journal of Computer Vision, Vol. 2, No. 3, pp. 251-282

[Wegner 1990] Peter Wegner, Concepts and Paradigms of Object-Oriented Programming, OOPS Messenger Vol 1 No 1, August 1990.

SECTION IV

Technical Papers

A. Physics-based and Low Level Vision

B. Stereo

C. Motion

D. Shape Recovery and Analysis

E. Object Recognition and Scene Analysis

F. Aerial Photo Matching and Interpretation

G. Active Vision and Visual Strategies

H. Robotics Applications

I. IU Software and Parallel Processing

Subsection A:

Physics-based and Low Level Vision

Perceptual Organization Without Edges

J. Brian Subirana-Vilanova and Kah Kay Sung
M.I.T. Artificial Intelligence Laboratory
Cambridge, MA, 02139

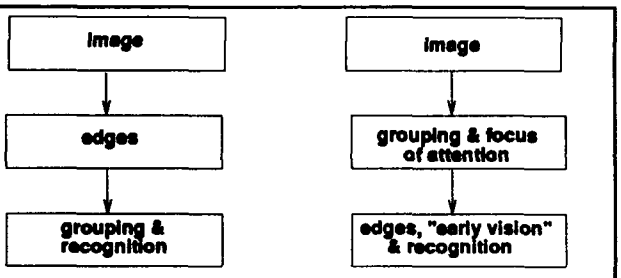


Figure 1: Two different views on the role of perceptual organization.

Abstract

Recent work in computer vision is based on the assumption that edge detection precedes grouping and object recognition. We (like others) argue against such an assumption and suggest that grouping precede the computation of discontinuities and most other early visual tasks.

We also present a perceptual organization scheme that works without edges. Our scheme includes a new ridge detector that is non-linear and *independent of scale.*

Results will be shown of a Connection Machine implementation of our scheme for perceptual organization (without edges) using color (the scheme is designed to work also for brightness and texture).

1 Introduction

Perceptual organization is a process that establishes what regions of the image come from one single object (of interest if possible), generally with very little detailed knowledge of the particular objects present. Recent work on computer vision has emphasized the role of edge detection and discontinuities in perceptual organization and recognition. This line of work stresses that edge detection should be done at an early stage on a brightness representation of the image, and segmentation and other early vision modules operate later on (see Figure 1 left). In this paper, we (like others) argue against such an approach and present a scheme that segments an image without finding brightness, texture, or color edges (see Figure 1 right). In our scheme discontinuities are found as a byproduct of the perceptual organization process.

Segmentation without edges is not new. Previous approaches fall into two classes. The first class is based on coloring or region growing [Hanson and Riseman 1978], [Horowitz and Pavlidis 1974], [Haralick and Shapiro 1985]. These schemes proceed by laying a few "seeds" in the image and then growing these until a complete region is found. The growing is done using a local threshold function. These growing schemes are limited in two ways: First, the growing function does not incorporate global factors which results in fragmented and superposed regions (see Figure 2). Second, there is no way to incorporate a priori knowledge on the shapes that we are looking for. For example, the Gestalt principles extensively used on current grouping algorithms have not been incorporated in existing coloring algorithms. In this paper we present a non-local perceptual organization scheme that uses no edges and which embodies gestalt principles such as symmetry, convexity and proximity. The second class of segmentation schemes which work without edges are based on volumetric descriptions of shapes. Examples include: [Badler and Bajcsy 1978], [Binford 1971], [Brooks, Russel and Binford 1979], [Brooks 1981], [Guzman 1968], [Pentland 1988] and [Waltz 1975]. These schemes make strong assumptions about the *three-dimensional* primitives used to describe the data. We believe that this is too strong a commitment to be a representation for late vision. It may be adequate for a restrictive set of recognition tasks but not in general. Our scheme conforms more closely to the data than these schemes and does not impose any three-dimensional interpretation of the data.

The scheme that we will present is an extension of [Subirana-Vilanova 1990] brightness-based perceptual organization scheme. Such scheme is based on a filter-based ridge detector which has a number of important problems that we will discuss. These include its dependence on scale and its sensitivity to curved shapes. This poses a number of additional constraints to be taken into account. Our analysis will lead us to a filter that we will show overcomes most of such problems. Such filter is based on a non-linear combination of two (otherwise linear) filters. Our scheme is designed to work for brightness, texture and color but our implementa-

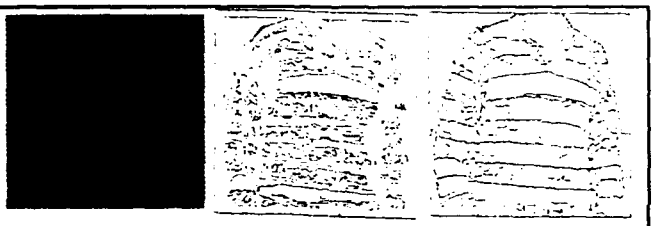


Figure 2: *Left:* An image of a shirt. *Center:* Original seeds for a region growing segmentation algorithm. *Right:* Final segmentation obtained using a region growing algorithm.

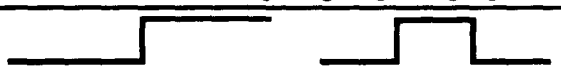


Figure 3: *Left:* Model of an edge. *Right:* Model of a ridge or box. Are these appropriate?

tion deals only with color. Color is an interesting case to study because it is a three-dimensional property, not one-dimensional like intensity. And therefore the extension of brightness based schemes to color is non-trivial.

We begin in the next section by listing several reasons for exploring non-edge based schemes and then present our approach. Results of a version of our scheme implemented on the Connection Machine will be shown.

2 In favor of regions

What is an edge? Unfortunately there is no agreed definition of edge. It can be defined in several related ways: as a discontinuity in a certain property, as something that looks like a step edge [Canny 1983] (see Figure 3), or by an algorithm (e.g. zero-crossings [Marr and Hildreth 1980]). Characterizing edges has proven to be difficult especially near corners, junctions and when there are edges at multiple scales, noise or transparent surfaces.

What is a region? Attempting to define regions bears problems similar to those encountered in the definition of an edge. Roughly speaking, it is a collection of pixels in an image that share a common property. In this context, an edge is the border of a region. But how can we find regions in images? We could proceed in a similar way as with edges, so that a region be defined (in one dimension) as a structure that looks like a box (see Figure 3). But this suffers from problems similar to the ones mentioned for edge detection.

Thus, regions and edges are two concepts closely related. It is unclear how we should represent the information contained in an image. As regions? As edges? Furthermore, independently of our choice, which structures should we try to recover first? Edges or regions? We believe that computer vision has over emphasized the early computation of discontinuities (whether brightness, texture or color discontinuities). Here are some reasons why exploring the computation of regions (without edges) may be a promising approach (see [Subirana-Vilanova and Sung 1991] for a more comprehensive discussion):

- There is psychological evidence that humans can recognize images with region information better than line drawings ([Cavanaugh 1991], but see also [Ryan and Schwartz 1956], [Biederman and Ju 1988]).
- Representations which maintain some region information such as the sign-bit of the zero crossings (instead of just the zero crossings themselves) are useful for perceptual organization.
- The performance of most rigid-object schemes is bounded by the complexity of the feature space used for exploring possible matchs. Additional region groups should reduce such complexity.
- Previous research on recognition has focused on rigid objects. Related grouping research has focussed on finding small sets of features with high likelihood of coming from the same object. For non-rigid objects this is not sufficient. Instead, it is necessary to group most of the features coming from a single object. We find it hard to believe that edge-features will be sufficient for bottom-up grouping in this case.
- Scale and stability are recognized as important problems. However, is it the stability and scale of an edge? or that of a region that we are interested in? Our scheme addresses stability in terms of objects (not edges). In addition, our scheme commits to one scale corresponding to the object of interest chosen by our scheme.

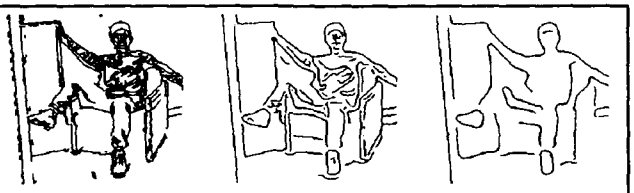


Figure 4: Edges computed at three different scales for an image of a person. Note that the results are notably different. Which scale is best?

3 Regions? What Regions?

In the last section we have set forth an ambitious goal: Develop a perceptual organization scheme that works on the image itself, without edges. But what constitutes a good region? What "class" of regions ought to be found? How can we know if our scheme is performing properly?

Our work is based on the observation that many objects in nature (or their parts) have a common color or texture, and are long, wide, symmetric and convex. This hypothesis is hard to verify formally, but it is at least true for a collection of common objects [Snodgrass and Vanderwart 1980] used in psychophysics. And as we will show, it can be used in our scheme yielding seemingly useful results. In addition, humans seem to organize the visual array using this type of principles as demonstrated by the Gestalt Psychologists. In fact, these were the starting point for much of the work in computer vision on perceptual organization for rigid objects. We use these same principles but in a different way: Without edges and with non-rigid shapes in mind.

4 Color, Brightness Or Texture?

The perceptual organization scheme presented in this paper includes color, brightness and texture. We decided to implement it on color first, without texture or brightness. Color based perceptual organization (without the use of other cues) is indeed possible for humans since two adjacent untextured surfaces viewed under isoilluminance can be segmented. (Although the human visual system has certain limitations in isoilluminant displays, e.g. [Cavanaugh 1987].) And, as we will discuss later in the paper, color is also useful when there are brightness changes.

Color is a perceived property of a surface that under normal conditions, depends mostly upon surface spectral reflectance and very little on the spectral characteristics of the light that enter our eyes. It is therefore useful for describing the material composition of a surface (independently of its shape and imaging geometry) [Rubin and Richards 1981]. Lambertian color is indeed uniform over most untextured physical surfaces, and is stable in shadows, and under changes in the surface orientation or the imaging geometry. In general it is more stable than texture or brightness.

It has long been known that the perceived color (or intensity) at any given image point depends on the light reflected from the various parts of the image, and not only on the light at that point. This is known as the simultaneous-contrast phenomena and has been known at least since E. Mach reported it at the beginning of the century. [Marr 1982] suggests that such a strategy may be used because one way of achieving some compensation for illuminance changes is by looking at differences rather than absolute values. According to this view, a surface is yellow because it reflects more "yellow" light than a blue surface, and not because of the absolute amount of yellow light reflected (of which the blue surface may reflect an arbitrary amount depending on the incident light).

The exact algorithm by which humans compute perceived color is still unclear. Our scheme only requires a rough estimate of color which is used to segment the image, see Figure 1. We believe that perceived color should be computed at a later stage by a process similar to the ones described in [Helson 1938], [Judd 1940], [Land and McCann 1971]. This model is in line with the ones presented in [Subirana-Vilanova and Richards 1991] and [Jepson and Richards 1991] which suggest that perceptual organization is a very early process which precedes most early visual processing.

In our images, color is entered in the computer as a "color vector" with three components: the red, green and blue channels of the video signal. Our scheme works mostly on color differences $\mathcal{S}_{\otimes}$ between pairs of pixels c and c_R. The difference that we used is defined in equation 1 and was taken from [Sung 1991] ($\otimes$ denotes the *vector cross product* operation) and responds very sensitively to color differences between similar colors [Sung 91].

$$\mathcal{S}_{\otimes}(c) = 1 - \frac{|c \otimes c_R|}{|c||c_R|} \qquad (1)$$

This similarity measure is a decreasing function with respect to the angular color difference. It assigns a maximum value of 1 to colors that are identical to the reference "ridge color", c_R, and a minimum value of 0 to colors that are orthogonal to c_R in the RGB vector space.



Figure 5: The similarity measure described in Equation 1 is illustrated here for an image of a person. *Left:* Image. *Center:* Similarity measure using as reference color the color of the pixel located at the intersection of the two segments shown. *Right:* Plot of the similarity measure along the long segment using the same reference color.

The discriminability of this measure can be seen intuitively by looking at the normalized image see Figure 5. The exact nature of this measure is not critical to our algorithm. What is important is that when two *adjacent* objects have different perceived color (in the same background) this measure is positive[1]. Many other measures have been proposed in the literature and could be incorporated in our scheme. What most color similarity measures have in common is that they are based on vector values and cannot be mapped onto a one-dimensional [Judd and Wyszecki 75] field[2]. This makes color perception different from brightness from a computational point of view since not all the one-dimensional techniques used in brightness images extend naturally to higher dimensions.

5 Previous work on perceptual organization without edges

The scheme that we present in this paper is an extension of Curved Inertia Frames (CIF), a brightness-based segmentation scheme presented in [Subirana-Vilanova 1990], which in turn is an extension of an *edge*-based perceptual organization scheme presented in the same paper. We choose this scheme for two reasons, first it is the only existing scheme that can compute global regions directly on the image without imposing a three-dimensional representation of the data. Second, we have been able to overcome a number of problems in the scheme so that it is useful for a large class of images.

[Subirana-Vilanova 1990]'s scheme proceeds in three

[1] Note that the perceived color similarity among arbitrary objects in the scene will obviously not correspond to this measure. Specially if we do not take into account the simultaneous-contrast phenomena

[2] Note that using the three chanels, red, green and blue independently works for some cases. However it is possible to construct cases in which it does not as when an object has two discontinuities, one in the red channel only and the other in one of the other two chanels only. In addition, the perceived similarity is not well captured by the information contained in the individual chanels alone but on the combined measure.

stages. In the first one, it computes two local measures at each point p for a number of orientations θ: the *inertia value* $\mathcal{I}(p,\theta)$ and the *tolerated length* $\mathcal{T}(p,\theta)$. These two local values are computed as the output of elongated gabor filters and are used to associate a *saliency measure* to *each* curve $C(t)$ in the image plane as defined in equation 2, were the curve is assumed to be parameterized between 0 and L, $\mathcal{I}(l)$ ($\mathcal{T}(t)$) is the inertia value (tolerated length) at the point with parameter l and with the orientation of the curve at that point, and ρ and α are suitable constants.

$$S_L = \int_0^L \mathcal{I}(l)\rho^{\int_0^l \frac{1}{\alpha \mathcal{T}(t)}dt}dl \qquad (2)$$

In the second stage, the scheme computes the skeleton which yields the maximum saliency using the network introduced by [Shashua and Ullman 1988]. In fact, the form of equation 2 closely matches what the network can compute. The inertia value and the tolerated length can be used in the second stage using other schemes such as [Kass, Witkin and Terzopoulos 88] and [Zucker, Dobbins and Iverson 89].

The scheme favors curves which are long, smooth (according to the associated tolerated length values) and central to the shape (i.e. which have high inertia values). This second stage yields the *skeleton sketch* a representation of the potential skeletons in the image. See [Subirana-Vilanova 1990], [Subirana-Vilanova 1991] for more details.

In the third stage, the scheme computes a succession of individual curves (or skeletons) and the corresponding perceptual groups by growing outward from the skeletons.

The extension of this scheme to color seems at first quite obvious: use elongated "vector" filters to compute the inertia values and the tolerated length. However, there are a number of important problems with such an approach. These will be discussed in the next section. One of the contributions of this paper is a new way of estimating these local parameters by defining a novel one-dimensional multi-scale color ridge detector that addresses the problems with previous approaches.

6 Problems in Finding Brightness Ridges

The one-dimensional version of the problem that we are trying to solve is to locate *ridges* in a signal. By ridge we mean something that looks like a pair of step edges (see Figure 3). A simple-minded approach is to find the edges in the image, and then look for the center of the two edges. This was the approach used in [Subirana-Vilanova 1990]. Another possibility is to design a filter to detect such a structure as in [Canny 1985], [Noble 1988]; this was the essence of the brightness based approach used in [Subirana-Vilanova 1990].

However, there are a number of problems with using such filters as estimators for ridge detection. These problems are not particular to [Subirana-Vilanova 1990]'s scheme, but are linked to the nature of ridges in real images. Some of these problems are in fact very similar for color and for brightness images. [Canny 1983] developed an optimal one-dimensional operator for edge detection which he then extended to two-dimensional images with remarkable performance.



Figure 6: *Left:* Plot with multiple steps. A ridge detector should detect three ridges. *Right:* Plot with narrow valleys. A ridge detector should be able to detect the different lobes independently of the size of the neighboring lobes.

[Canny 1983] used a similar methodology to find ridges. His model of a ridge was similar to the one shown in Figure 3. This is a limited model since ridges in images are not well suited to it. Perhaps the most evident reason why such model is not realistic is the fact that it is tuned to a particular scale. However, in most images, ridges appear at multiple and unpredictable scales. This is not so much of a problem in edge-detection as we have discussed in the previous sections, because the edges of a wide range of images can be assumed to have "a very similar scale". Thus, Canny's ridge detector works only on images where all ridges are of the same scale. This is true in the text images shown in [Canny 1983] (see also Figure 8) and in the images used by [Subirana-Vilanova 1990].

Therefore, an important feature of a ridge detector is its scale invariance. We now summarize a number of important features that a ridge operator should have (see Figure 6):

- *Scale:* See previous paragraph.
- *Non-edgeness:* The filter should give no response for a step edge. This property is violated by [Canny 1985].
- *Multiple steps:* The filter should also detect small steps. These are frequent in images for example when an object is occluding the space between two other objects. This complicates matters in color images because the surfaces are defined by vectors no just scalar values.
- *Narrow valleys:* The operator should also work in the presence of multiple ridges even when they are separated by small valleys.
- *Noise:* As with any operator that is to work in real images, tolerance to noise is necessary.
- *Localization:* The output of the ridge-detector should be higher in the middle of the ridge than on the sides.
- *Strength:* The strength of the response should be somehow correlated with the strength of the perception of the ridge by humans.
- *Large scales:* Large scales should receive higher response. This is a property used by [Subirana-Vilanova 1990]'s scheme and is important because it embodies the preference for large objects.

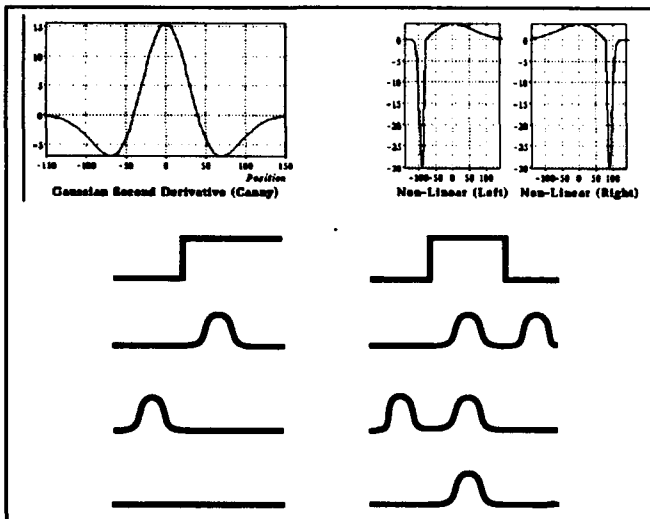


Figure 7: *Top Left:* Gaussian second derivative, an approximation to Canny's optimal ridge detector. *Top Center and Right:* Individual one-dimensional masks used by our operator. *Bottom:* Response for an edge (left) and a ridge (right) for: the left operator of our ridge detector (second row), the right operator (third row) and the combined minimum of the two responses (fourth row). In all cases, sigma is roughly the size of the ridge. This is a qualitative demonstration. It illustrates why combining the responses of two filters yields a positive response for a ridge but no response for an edge.

7 A color ridge detector

In the previous section we have outlined a number of properties that we would like our ridge-detector to have. As we have mentioned, the Canny ridge-detector fails because it can not handle multiple scales. A naive way of solving this problem would be to apply the Canny ridge detector at multiple scales and define the output of the filter at each point as the response at the scale which yields a maximum value. This filter would work in a number of occasions but has the problem of giving a response for step edges (since the ridge-detector at any single scale responds to edges so will the combined filter do).

One can prevent the response of edges by splitting Canny's ridge operator into two pieces, one for each edge, and then combining the two responses by looking at the minimum of the two responses. Figure 7 outlines the philosophy of our approach. Figure 8 illustrates how our filter behaves according to the different criteria outlined before. The Figure also compares our filter with that of the second derivative of a Gaussian, which is similar to the filter used by Canny. There are a number of potential candidates within this framework such as splitting Canny filter by half, using two edge detectors and many others. We tried a number of possibilities on the Connection Machine on a real and on a synthetic image with varying degrees of noise. Table 1 presents the filter which gave a response most similar to the inertia values and the tolerated length that one would obtain using the formulas for the corresponding edges as described in [Subirana-Vilanova 1990].

Var	Description/Expression
$\mathcal{P}_{max}$	Gradient penalization coeff: (3)
F_s	Filter Side Lobe size coeff: (1/8)
F_c	Local Neighborhood size coeff: (1/8)
$g(x)$	Color gradient at location x.
g_{max}	Maximum color gradient in image.
σ	Size of Main Filter Lobe.
σ_s	Size of Side Filter Lobe: $F_s\sigma$
σ_c	Reference Color Neighborhood: $F_c\sigma$
$c(x)$	Color vector at x: $[\mathcal{R}(x)\ \mathcal{G}(x)\ \mathcal{B}(x)]^T$
$c_n(x)$	Normalized Color at x: $c(x)/\lvert c(x)\rvert$
$c_r(x)$	Refr. Color: $\int_{-\sigma_c}^{\sigma_c} \frac{1}{\sqrt{2\pi}\sigma_c} e^{-\frac{r^2}{2\sigma_c^2}} c_n(x+r)\,dr$
$\mathcal{F}_L(r)$	Left and Right Halves of Filter: $\frac{r+\sigma}{\sigma^3\sqrt{2\pi}} e^{-\frac{(r+\sigma)^2}{2\sigma^2}}$ $-\sigma < r < \sigma$; $\frac{r+\sigma}{\sigma_s^3\sqrt{2\pi}} e^{-\frac{(r+\sigma)^2}{2\sigma_s^2}}$ $-(\sigma+\sigma_s) < r < -\sigma$; 0 otherwise
$\mathcal{F}_R(r)$	$\mathcal{F}_L(-r)$
$\mathcal{I}_L(x)$	Inertia from Left and Right Halves: $\int_{-(\sigma+\sigma_s)}^{\sigma} S_{\otimes}(c_r(x), c_n(x+r))\mathcal{F}_L(r)\,dr$
$\mathcal{I}_R(x)$	$\int_{-\sigma}^{\sigma+\sigma_s} S_{\otimes}(c_r(x), c_n(x+r))\mathcal{F}_R(r)\,dr$
$\mathcal{I}_\sigma(x)$	Inertia at location x (Scale σ) $\min(\mathcal{I}_L(x), \mathcal{I}_R(x))\ \frac{\sqrt{\sigma}}{(1+\mathcal{P}_{max}\frac{g(x)}{g_{max}})^2}$
$\mathcal{I}(x)$	Overall Inertia at location x: $\forall\sigma\ \max(\mathcal{I}_\sigma(x))$

Table 1: Steps for Computing Directional Inertias

Our approach uses two filters (see profile in Figure 7), each of which looks at one side of the ridge. The output of the combined filter is the minimum of the two responses. Each of the two parts of the filter is asymmetrical, reflecting the fact that we expect the object to be uniform (central part of the filter is large), and that we do not expect that a region of equal size be adjacent to the object (lateral part of the filter is small). In other words, our ridge detector is not fooled by narrow valleys.

The extension to color is tricky because there is no clear notion of what is positive and what is negative in vector quantities. We solve this problem by adaptively defining a reference color at each point as the weighted average color over a small neighborhood of the point (about eight times smaller than the scale of the filter). Thus, this reference color will be different for different points in the image. Then, scalar deviations from this reference color are computed as defined in section 4.

8 Results

We have tried our filter extensively, Figure 8 shows that the output of our filter is better than that of a sec-

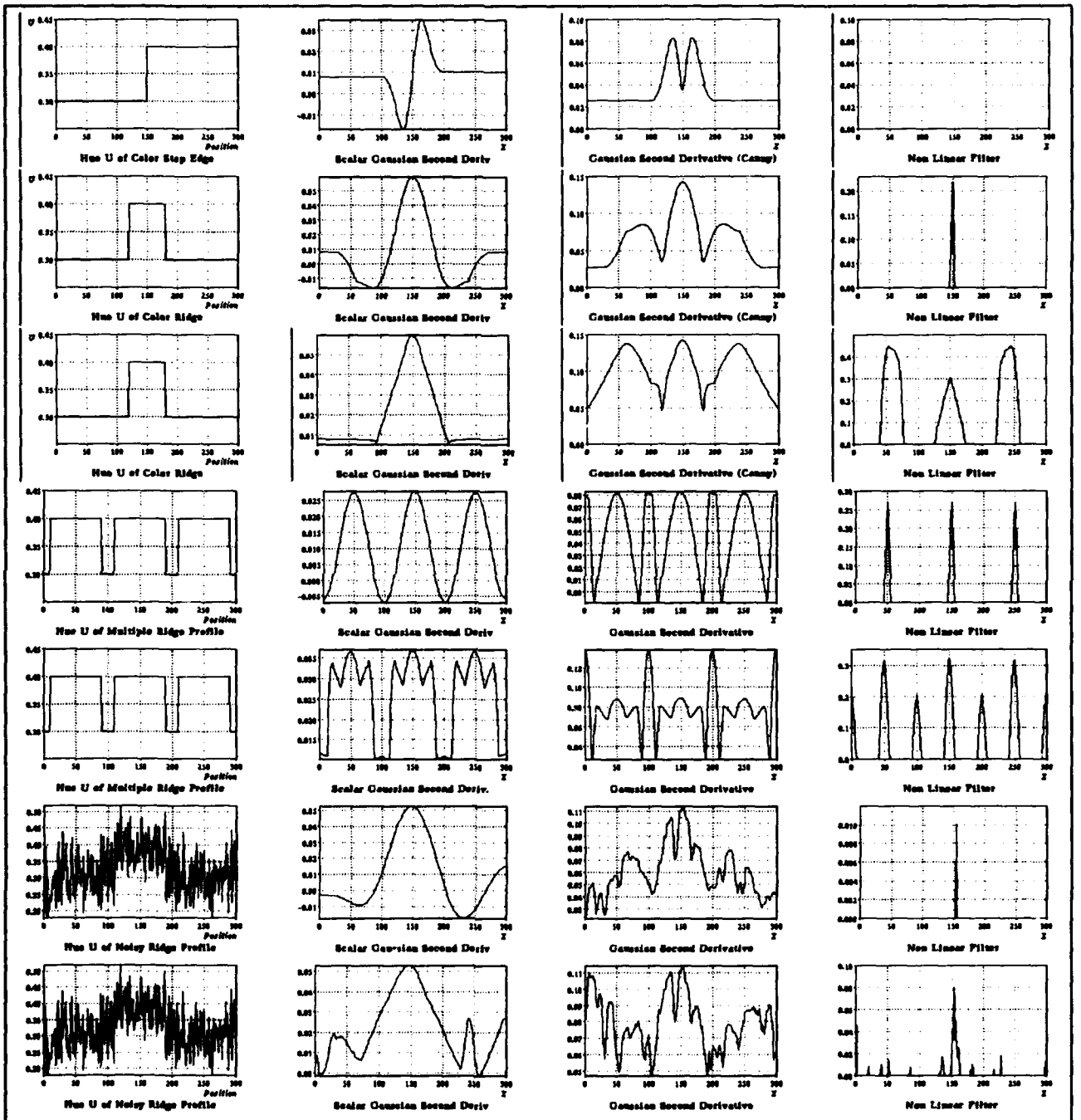


Figure 8: *First column:* Different input signals. *Second column:* Output given by second derivative of the gaussian. *Third column:* Output given by second derivative of the gaussian using reference color. *Fourth column:* Output given by our ridge detector. The *First, Second, Fourth and Sixth rows* are results of a single scale filter application. The *Third, Fifth and Seventh rows* are results of a multiple scale filter application.



Figure 9: Two images. *Left:* Ribbons image. *Right:* Blob image. See inertia surfaces for the ribbon image in Figure 10. Note that our scheme recovers the blob at the right scale, without the need of specifying the scale.

ond derivative of a Gaussian even with the notion of a reference color. First, our filter localizes all the ridges for a single ridge, for multiple or step ridges and for noisy ridges. The second derivative of the Gaussian instead, fails under the presence of multiple or step ridges. Second, the scale chosen by our operator matches the underlying data closely while the scale chosen by the second derivative of the Gaussian does not match the underlying data. This is important because the scale is necessary to compute the Tolerated Length which is used in the second stage of our scheme to find the Curved Inertia Frames of the image. And third, our filter does not respond to edges while the second derivative of the Gaussian does.

In the previous paragraph we have discussed the one-dimensional version of our filter. The filter can be applied to two-dimensional images at different orientations. Figure 10 shows such output (aka inertia surfaces) of our filter for four images. The two-dimensional version of the filter can be used with different degrees of elongation. In our experiments we used one pixel width to study the worst possible scenario. An elongated filter would smooth existing noise; however, large scales are not good because they smooth the response near discontinuities and in curved areas of the shape (this can be overcomed by using curved filters [Malik and Gigus 1991]).

The inertia surfaces and the tolerated length are the output of the first stage of our scheme. In the second stage we use these to compute the Curved Inertia Frames (see [Subirana-Vilanova 1990]) as shown in Figures 11 and 12. These skeleton representation is used to grow the corresponding regions by a simple region growing process which starts at the skeleton and proceeds outward. This process is very stable because it can use global information provided by the frame such as the average color or the expected size of the enclosing region. See Figures 11 and 12. for some examples of the regions that are obtained. Observe that the shape of the regions is accurate, even at corners or junctions.

Thus, each stage of our scheme has been tested extensively. Note that each region can be seen as an individual test since the computations performed within it are independent of those performed outside it.

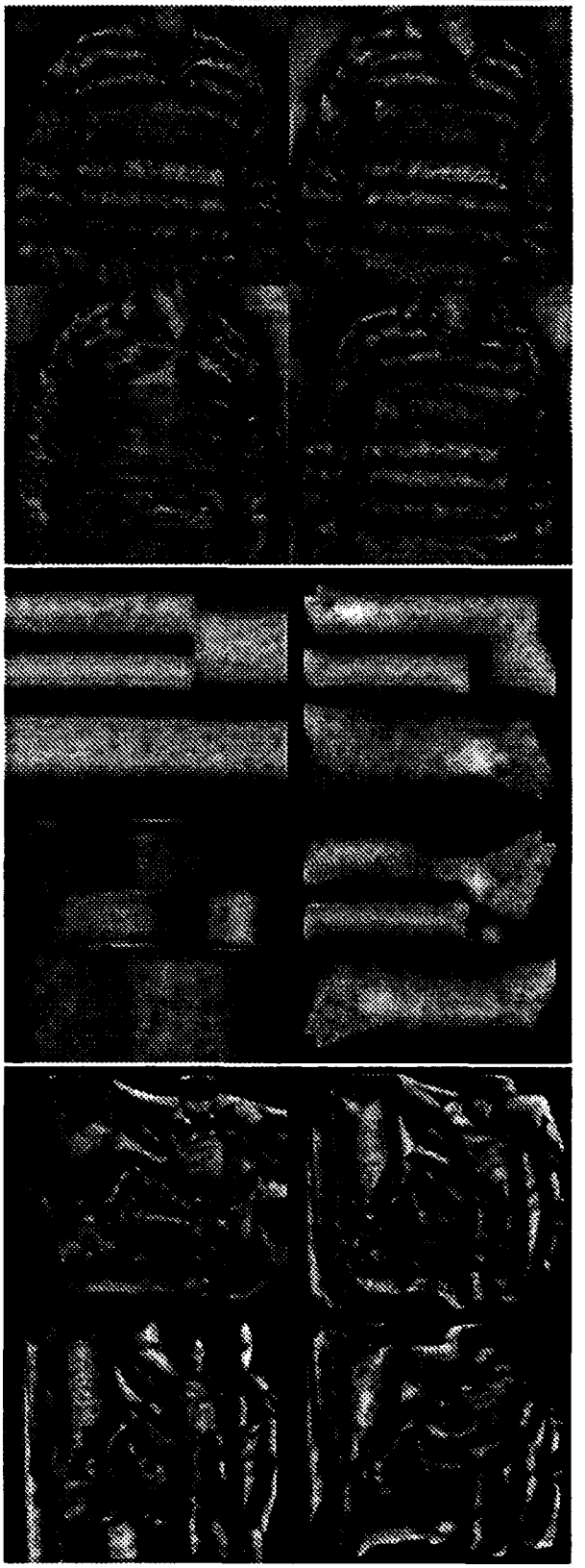


Figure 10: Inertia surfaces for Three images at four orientations (clockwise 12, 1:30, 3 and 4:30). Note that *exactly* the same lisp code (without changing the parameters) was used for all the images. *Top:* Shirt image. *Middle:* Ribbon image. *Bottom:* Person image.

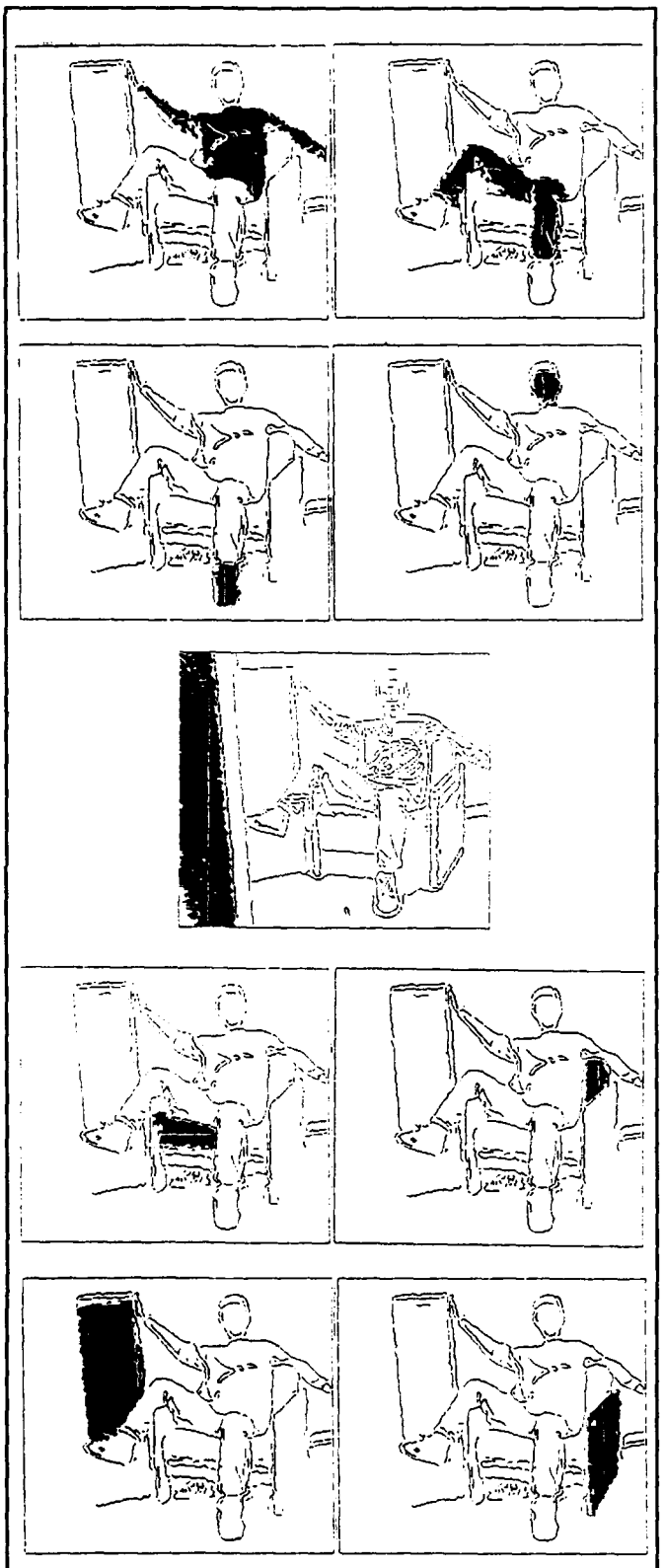


Figure 11: Regions obtained for the person image. The white curves are the Curved Inertia Frames from which the regions were recovered.

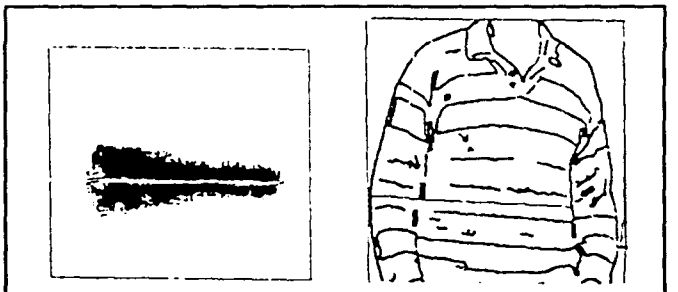


Figure 12: *Left:* Blob obtained using our scheme in the blob image. *Right:* Most salient Curved Inertia Frame obtained in the shirt image. Note that our scheme recovers the structures at the right scale, without the need of changing any parameters.

9 Brightness and Edges

We have implemented our scheme on the Connection Machine for color. The scheme can be extended naturally to brightness and texture (using filter-based approaches applied to the image, see [Turner 86], [Malik and Perona 89] and [Bovik, Clark and Geisler 1990]). The more cues a system uses, the more robust it will be. In fact, brightness is crucial in some situations because brightness boundaries do not always come together with color boundaries (e.g. cast shadows).

But, should these different schemes be applied independently? Consider a situation in which a surface is defined by an isoluminant color edge on one side and by a brightness edge (which is not a color edge) on the other. Our scheme would not recover this surface because the two sides of our filter would fail (on one side for the brightness module and on the other for the isoilluminant one). We believe that a combined filter should be used to obtain the inertia values and the tolerated length in this case. The second stage would then be applied only to one set of values. Instead of having a filter with two sides, our new combined filter should have four sides. Two responses on each side, one for color $R_{c,i}$ and one for brightness $R_{b,i}$, the combined response would then be $min(max(R_{b,left}, R_{c,left}), max(R_{b,right}, R_{c,right}))$.

In addition, there are a number of situations in which edges are clearly necessary as in line-drawings or in the Kanizsa figures. Surfaces are also perceived but it is hard to believe that discontinuities do not play a role in this case.

10 Discussion

Our scheme solves the problem of finding different regions by looking at the large structures one by one. The larger structures are the first ones in being recovered, this cuts small structures that are covered by larger structures into different parts. This embodies the constraint that larger structures tend to be perceived as occluding surfaces [Petter 1956]. (See Figure 13).

As mentioned in [Subirana-Vilanova 1991], the emphasis of C.I.F. is towards finding large structures. However, this may be misleading as evidenced by Figure 14. The interesting structure is not composed by individual elements that pop-out in the background. Instead, in



Figure 13: Large shapes occlude small ones. From [Kanizsa 1979].

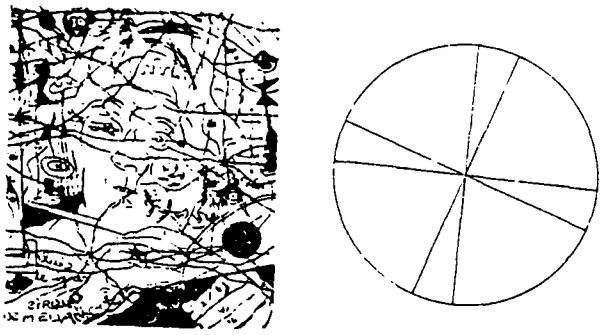


Figure 14: Small structures, whether edges or regions are sometimes more salient. *Left:* Drawing of Miró. *Right:* From [Rock 1984].

this case, what seems to capture our attention can be described as what is not large. That is, looking for the large structures and finding what is left would recover the interesting structure as if we where getting rid of the background. It is unclear though, if this observation would hold in general. Future research is necessary.

11 What's New

In this paper we have argued that early visual processing should seek representations that make regions explicit, not just edges. Furthermore, we have argued that region representations should be computed directly on the image (i.e. not directly from discontinuities). These suggestions can be taken further to imply that an attentional "coordinate" frame (which corresponds to one of the perceptual groups obtained) is imposed in the image prior to constructing a description for recognition. See also [Subirana-Vilanova and Richards 1991].

We have provided evidence in favor of our suggestions by considering a number of issues related to edge-detection such as stability, scale, data representation, human perception, perceptual organization, junctions and corners.

Our model suggests that vision should start by computing a set of features all over the image (corresponding to the inertia values and the tolerated length). This can be thought of as "smart" convolutions of the image with suitable filters plus some simple non-linear processing. In fact, recently, remarkable success has been achieved by filter-based approaches to texture [Knuttson and Granlund 1983], [Turner 1986], [Fogel and Sagi 1989], [Malik and Perona 1989], [Bovik, Clark and Geisler 1990], stereo [Kass 1983], [Jones and Malik 1990] brightness edge detection [Canny 1986], [Morrone, Owens and Burr 1987, 1990], [Freeman and Adelson 1990] and motion [Heeger 1988]. (See also [Abramatic and Faugeras 1982], [Marrone and Owens 1987]). Our proposal differs from theirs in that we use the output of the filters to look for regions, not discontinuities.

This has been the motivation for designing a new non-linear filter for ridge-detection. Our ridge detector has a number of advantages over previous ones since it selects the appropriate scale at each point in the image, does not respond to edges, can be used with brightness as well as color data, is tolerant to noise[3] and can handle narrow valleys and multiple steps.

The resulting scheme can segment an image without making explicit use of discontinuities and is computationally efficient on the Connection Machine (takes time proportional to the size of the image). Our scheme works properly near junctions and corners because it uses global information. The performance of the scheme can in principle be attributed to a number of intervening factors; but we believe that the critical aspect of the scheme (and one of the contributions of this paper) is our ridge-detector. Running the scheme on the edges or using simple gabor filters would not yield comparable results. The effective use of color makes the scheme very robust but we believe that comparable results would be obtained on brightness or texture data.

References

[1] J-F. Abramatic and Faugeras O. Sequential convolution techniques for image filtering. *IEEE ASSP*, 30(1):1–10, 1982.

[2] N. Badler and R. Bajcsy. Three-dimensional representations for computer graphics and computer vision. *CG*, 12:153–160, 1978.

[3] T.O. Binford. Visual perception by a computer. In *Proc. of the IEEE S. C., Miami*, 1971.

[4] Bovik, Clark, and Geisler. Multichannel texture analysis using localized spatial filters. *IEEE*, PAMI-12:56–65, 1990.

[5] R.A. Brooks. Symbolic reasoning among 3-D models and 2-D images. *AI*, 17:285–348, 1981.

[6] R.A. Brooks, G. Russell, and T. Binford. The acronym model based vision system. In *Proc. IJCAI*, pages 105–113, 1979.

[7] J. Canny. A computational approach to edge detection. *IEEE*, PAMI-8:679–698, 1986.

[8] P. Cavanagh. Reconstructing the third dimension: interactions between color, texture, motion, binocular disparity, and shape. *CVGIP*, 37:171–195, 1987.

[9] D. Clemens. *Region-based feature interpretation for recognizing 3D models in 2D images.* PhD, MIT 1991.

[10] B.G. Cumming, A.C. Hurlbert, E.B. Johnson, and A.J. Parker. Effects of texture and shading on the kde. In *ARVO Vol 32, NO. 4*, page 1277 20814-3928, 991.

[3] We have verifyed this empirically but we have not performed a detailed study of the effect of noise in our filter. An improved version of the filter may be obtained from such exercise.

[11] I. Fogel and D. Sagi. Gabor filters as texture discriminators. *B.C.*, 61:103–113, 1989.

[12] W.T. Freeman and E. H. Adelson. Steerable filters for early vision, image analysis and wavelet decomposition. In IEEE Computer Society, editor, *Third ICCV*, pages 406–415, 1990.

[13] W.E.L. Grimson. *Object Recognition By Computer. The Role Of Geometric Constraints.* MIT Press, 1990.

[14] A. Guzman. Decomposition of a visual scene into three-dimensional bodies. In *Proc. AFIPS 1968 Computer Conference*, pages 291–304, 1968.

[15] A. R. Hanson and E. M. Riseman. Segmentation of natural scenes. *CVS.*

[16] R.M. Haralick and L.G. Shapiro. Image segmentation techniques. *CVGIP*, 29:100–132, 1985.

[17] D. Heeger. Optical flow from spatiotemporal filters. In *First ICCV*, pages 181–190, 1988.

[18] H. Helson. Fundamental principles in color vision. *JEP*, 23:439–471, 1938.

[19] S. L. Horowitz and T. Pavlidis. Picture segmentation by a direct split and merge procedure. pages 424–433, August 1974.

[20] D.P. Huttenlocher and P. Wayner. Finding convex edge groupings in an image. TR 90-1116, Comp. Sc., Cornell U., 1990.

[21] D.W. Jacobs. Grouping for recognition. A.I. TR 1117, MIT AI, 1989.

[22] A. Jepson and W. Richards. Integrating vision modules. *IEEE S. C.*, 1991. Sp. Iss. Assimilation. Editor: A. Jain. To appear.

[23] D. Jones and J. Malik. Computational stereopsis–beyond zero-crossings. *Invest. Ophtalmol. Vis. Sci. (Supp.)*, 31(4):529, 1990.

[24] D. B. Judd and G. Wyszecki. *Color in Business, Science and Industry (Third Edition).* Wiley, 1975.

[25] D.B. Judd. Appraisal of land's work on two-primary color projections. *JOSA*, 50:254–268, 1940.

[26] G. Kanizsa. *Organization in Vision.* Praeger, 1979.

[27] M. Kass. Computing visual correspondence. In *From Pixels to Predicates*, pages 78–92. Ablex Pub. Co., 1986.

[28] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, pages 321–331, 1988.

[29] H. Knuttson and G.H. Granlund. Texture analysis using two-dimensional quadrature filters. In *Workshop on Comp. Arch. for Pattern Analysis and Image Database Management*, pages 206–213, IEEE Comp. Soc., 1983.

[30] D.G. Lowe. *Perceptual Organization and Visual Recognition.* PhD thesis, Stanford U., 1984.

[31] D.G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[32] J.V. Mahoney. Image chunking: defining spatial building blocks for scene analysis. A.I. TR 980, MIT AI 1987.

[33] J. Malik and Z. Gigus. A model for curvilinear segregation. *Invest. Ophtalmol. Vis. Sci. (Supplement)*, 32(4):715, 1991.

[34] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America*, 7(5):923–932, 1990.

[35] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* W.H. Freeman and Company, 1982.

[36] D. Marr and E. Hildreth. Theory of edge detection. *Proc. of the Royal Soc. of London*, B(207):187–217, 1980.

[37] J.L. Marroquin. Human perception of structure. Master's thesis, MIT, 1976.

[38] M.C. Morrone and D.C. Burr. A model of human feature detection based on matched filters. In P. Dario and G. Sandini, editors, *Robots and biological systems.* Academic Press, 1990.

[39] A. Pentland. Parallel part segmentation for object recognition. T.R. 108, Vis. Sc., Media Laboratory, MIT,1988.

[40] G. Petter. Nuove ricerche sperimentali sulla totalizzazione percepttiva. *Rivista di psicologia*, 50:213–227, 1956.

[41] I. Rock. *Perception.* Scientific american library, 1984.

[42] John M. Rubin and W. A. Richards. Colour vision and image intensities: When are changes material. TR AIM–631, MIT AI 1981.

[43] A. Sha'ashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. *Proc. First ICCV*, pages 321–327, 1988.

[44] J.G. Snodgrass and M. Vanderwart. A standarized set of 260 pictures: Norms for name agreement, image agreement, familiarity, and visual complexity. *JEP: HPP*, 6(2):174–215, 1980.

[45] J.B. Subirana-Vilanova. The skeleton sketch: Finding salient frames of reference. In *IU, 1990*, pages 399–414. Morgan and Kaufman, 1990. Also in the Proc. of ICCV'90.

[46] J.B. Subirana-Vilanova. In preparation. 1991.

[47] J.B. Subirana-Vilanova and W. Richards. Perceptual organization, figure-ground, attention and saliency. MIT AI memo 1218, 1991.

[48] J.B. Subirana-Vilanova and K.K. Sung. In preparation.

[49] K. Sung. Forthcoming. Master's thesis, Massachusetts Institute of Technology, 1991.

[50] M. Turner. Texture discrimination by gabor functions. *Bio. Cyb.*, 55:71–82, 1986.

[51] D.L. Waltz. Understanding line drawings of scenes with shadows. In P. Winston, editor, *The Psychology of Computer Vision.* McGraw-Hill, 1972.

[52] A.P. Witkin and J.M. Tenenbaum. On the role of structure in vision. In J. Beck, B. Hope, and A. Rosenfeld, editors, *Human and Machine Vision.* Acad. Press, 1983.

[53] S. Zhong and S. Mallat. Compact image representation from multiscale edges. In IEEE, *Third ICCV*, pages 406–415, 1990.

[54] S.W. Zucker, A. Dobbins, and L. Iverson. Two stages of curve detection suggest two styles of visual computation. *Neural Computation*, 1(1):68–81, 1989.

Contour Matching Using Local Affine Transformations

Ivan A. Bachelder and Shimon Ullman
MIT Artificial Intelligence Laboratory
Cambridge, MA, 02139

Abstract

Visual processing tasks often require the matching of contours in two images. Examples include determining optical flow, matching features for alignment-based object recognition, and finding correspondence for long-range and apparent motion. We propose a scheme for matching partially constrained contours in two images using local affine transformations. This new scheme is motivated in part by existing ideas for both recovering optical flow and matching features for object recognition. The scheme approximates the structure of orthographically projected contours with planar patches. Each patch is determined using oriented elliptical Gaussian neighborhoods that smoothly integrate information over proximally connected contours at several spatial scales. At the largest scale satisfying available constraints a minimal solution mechanism employs a modified general pseudoinverse to predict matches that are closest to the simplest purely translational correspondence. This realization avoids previously encountered difficulties that are typically caused by oversimplifying assumptions regarding contour transformations, lack of efficient implementation, and instability due to singular conditions. Additional advantages include the ease with which the proposed scheme can incorporate the influence of specially matched features. These properties are verified by simulation results obtained on noisy synthetic and natural imagery.

1 Introduction

At the heart of many visual processing tasks lies the determination of correspondence between features in two images. For example, correspondence is at least implicitly required in stereopsis, structure from motion, and object recognition. Often the matching is underconstrained because the features can be matched in more than one way. Furthermore, the search space for possible matches may be prohibitively large. Assumptions, usually derived from regularities of the physical world, are therefore necessary to constrain matching solutions uniquely. For example, in stereopsis one can constrain the search to small areas along an epipolar line by exploiting uniqueness, continuity, coarse to fine approximations, and probabilistic assumptions about disparity [23, 24, 28].

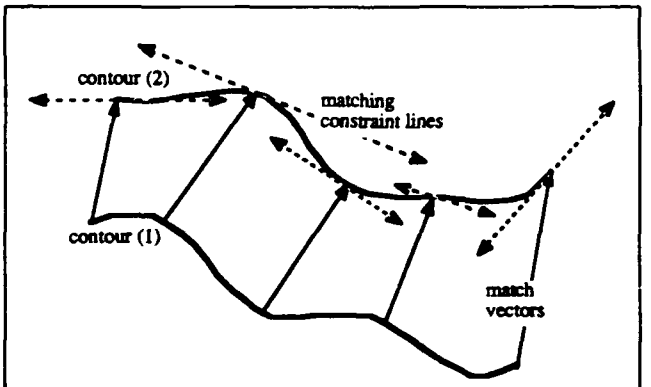


Figure 1: *To find a point-to-point correspondence between contours in two images we are given* **constraint lines** *for several points in the first image. Each constraint line narrows down the match a point in the first image to a line in the second.*

We address the general problem of matching partially constrained contours in a pair of images. As illustrated by Figure 1, we are given: (1) an image containing a set of contours; (2) a second image obtained by arbitrarily and independently displacing, stretching, rotating, scaling, and distorting these contours; and, (3) matching *constraint lines* for several points along the first set of contours constraining the match for each of these points to a line in the second image. These three sources of information are application specific. The contour images may, for instance, be obtained by applying an edge finding algorithm to either sampled time-varying imagery or two different views of a scene. Furthermore, the constraint lines accompanying these two images can often be estimated for many constrained contour matching applications.

One such application is the measurement of visual motion. In determining optical flow along isobrightness contours, the tangential component of motion must be constrained by additional assumptions due to the *aper-*

ture problem [10, 17, 1, 14]. Because local measurements of motion capture the component of image velocity only in the direction normal to isobrightness contours, the match for a point in one image frame is constrained to a line in the next image frame.

Another problem requiring the matching of contours arises in alignment-based methods for object recognition. In these methods, global transformations between model and object views are compensated for by a normalization stage, which aligns the two views and allows subsequent comparison. These views can be represented by contours. Though a major benefit of alignment approaches is the avoidance of exponential search by compensating for possible transformations before verification [18], alignment schemes require that object and model features be matched at some stage (For a broader overview of the alignment approach, see [31, 18, 6].) Assuming that model and object views can be roughly aligned using global image properties or several pre-matched "anchor points" [18], we claim that heuristic constraint lines can be estimated for each object contour point by the tangent at the closest model contour point [4]. (The main inaccuracies in this method will occur for constraint lines at high curvature points, which are simply ignored in the matching process.) A similar method can be used for finding correspondence for long-range and apparent motion.

This paper describes a new scheme for solving the constrained contour matching problem using local affine transformations. First, we briefly review several previous methods for both determining optical flow and finding matches for object recognition. Second, we describe the new scheme, which reworks some of the previous ideas into a robust, accurate, and efficient framework. Following the full development of this scheme, we discuss simulation results obtained on noisy synthetic and natural imagery.

2 Previous Matching Ideas

In computing optical flow, several methods have assumed that the motion of patterns can be described at least locally by pure translation [21, 27, 1]. However, while this assumption accurately tracks objects in the case of observer motion, a more general assumption is necessary to account for objects undergoing combinations of translation with three-dimensional rotation, scaling, and deformation. Other methods find the velocity field with the least variation that is consistent with local motion measurements [17, 25, 2, 11, 19]. When implemented along contours [12, 13, 26] such schemes can avoid smoothing over sharp discontinuities in the motion field. However, they usually require slowly converging iterative methods, such as the conjugate gradient method, to efficiently recover a global solution constrained by hundreds of linear equations [13]. A third approach is the assumption that objects in motion can be described by planar patches [33]. In this approach the second order terms of the Taylor series expansion of optical flow are determined by satisfying the normal flow constraints within fixed image neighborhoods. However, because the size of these neighborhoods remains fixed, highly non-planar segment boundaries cannot be accurately described. Recent affine motion models that include methods for selecting neighborhood sizes using Gaussian scale pyramids require iteration [8, 7]. Finally, a major drawback of all these methods is the existence of singular conditions for which no unique solution exists. Solutions typically become highly unstable (sensitive to noise) as objects approach such configurations. For example, the motion of near-linear contours is ambiguously interpreted by all of these methods.

In object recognition, several matching schemes employ correlation measures or examine angles and intersections between contours. Such methods either assume perfect alignment, or conduct a complete search through feature space. They cannot deal with un-parameterized distortions, or normalization inaccuracies prevalent in a roughly aligned contour image. They also typically lack efficient implementation. Recently, a number of methods have been devised for comparing contours using affine invariant curvature [9], arc length [3], or moments [15]. The main drawback of these methods is that they are global measures, applicable only to complete, unoccluded curves. Furthermore, as the objects producing the contour image becomes less planar, the affine invariance becomes less valid for describing transformations. A second disadvantage is that the calculation of these affine invariant quantities requires a high degree of differentiation along contours, implying both that contour tracing, contour thinning, and contour enhancement be performed in order to ensure a unique path along the contour, and that contour smoothing be performed in order to avoid large errors in the differentiation due to noise. Most importantly, though, is that all of these recognition techniques are used primarily as verification steps taken after a matching has already been hypothesized by exponential search or inaccurate heuristics. This is also true of other affine matching techniques [20, 32].

3 The Proposed Solution

Despite the inadequacies of the previous attempts, we selectively incorporate several of these ideas to solve the constrained contour matching problem. In addition, new thought is given to many key issues, such as finding the spatial extent over which these ideas are valid. To address these issues, we develop a scheme for matching constrained contours that is based on local affine transformations. It constrains the matching by assuming that contours are the orthographic projections of locally coplanar points, thereby reducing the recovery of correspondence to a local, linearly constrained, non-iterative calculation.

First, the transformations of contours within local image neighborhoods are assumed to be affine. We implicitly find the affine transformation through a least squares fit to available match constraint lines and pre-matched points. Second, local neighborhoods are constructed so that constraint line information is smoothly integrated primarily over proximal connected contours. This is accomplished by using elliptical Gaussian neighborhoods oriented along the contour. Third, we consider several neighborhoods of differing sizes simultaneously for

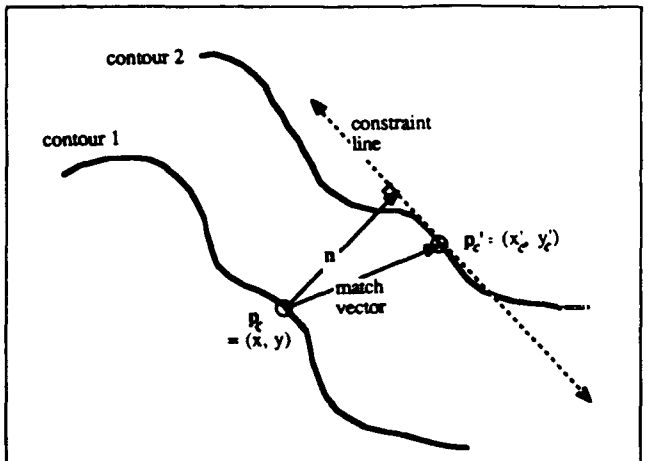


Figure 2: *The unknown match point, $\mathbf{p}'_c$, on contour 2 for $\mathbf{p}_c$ on contour 1 is assumed to lie along a known constraint line described by a constraint normal, $\mathbf{n}$.*

a given point. From these neighborhoods, we choose the largest neighborhood within which an affine transformation can accurately satisfy available constraints. Finally, a stable, unique solution is guaranteed for the chosen neighborhood by using a modified general pseudoinverse to find, subject to the constraints, the matches that deviate the least from the smallest purely translational correspondence.

4 Matching Globally Planar Contours

Let us first assume that the transformation between two image contours can be described by a *global* affine mapping. That is, each contour point $\mathbf{p}_c = (x_c, y_c)$ in the first image maps to contour point $\mathbf{p}'_c = (x'_c, y'_c)$ in the second image by the linear equation

$$\mathbf{p}'_c = A\mathbf{p}_c + \mathbf{t} \tag{1}$$

where the 2×2 matrix A accounts for the two-dimensional shearing, scaling, and rotation in the image plane about the global origin, $\mathbf{p}_o = (0,0)$, and the vector $\mathbf{t}$ accounts for two-dimensional translation in the image plane. Next, consider the contours in Figure 2. Suppose that the exact location of $\mathbf{p}'_c$ is unknown, but that it lies along some known *constraint line* in the image. If $\mathbf{n}$ is the perpendicular from $\mathbf{p}_c$ to the constraint line, and $\hat{\mathbf{n}}$ is the unit normal to the constraint line, then the distance from the constraint line is

$$\hat{\mathbf{n}}^T(\mathbf{p}_c + \mathbf{n} - \mathbf{p}'_c) = 0. \tag{2}$$

Substituting (1) into (2) yields

$$(A\mathbf{p}_c)^T\hat{\mathbf{n}} + \mathbf{t}^T\hat{\mathbf{n}} = \mathbf{p}_c^T\hat{\mathbf{n}} + |\mathbf{n}|. \tag{3}$$

Therefore, we obtain one linear equation constraining the six parameters of the affine transformation for each point. Let us represent the six affine parameters as a vector

$$\mathbf{a} = [\ A_{00} \quad A_{10} \quad A_{01} \quad A_{11} \quad t_x \quad t_y\ ]^T. \tag{4}$$

Then for each contour point $\mathbf{p}_{c_i}$ with constraint normal $\mathbf{n}_i$ equation (3) can be rewritten as

$$\mathbf{c}_i\mathbf{a} = d_i \tag{5}$$

where

$$\mathbf{c}_i = [\ \hat{n}_{i_x}x_{c_i} \quad \hat{n}_{i_x}y_{c_i} \quad \hat{n}_{i_y}x_{c_i} \quad \hat{n}_{i_y}y_{c_i} \quad \hat{n}_{i_x} \quad \hat{n}_{i_y}\ ] \tag{6}$$

$$d_i = \mathbf{p}_{c_i}^T\hat{\mathbf{n}}_i + |\mathbf{n}_i|. \tag{7}$$

Thus, for a system of k constrained points and associated equations,

$$C\mathbf{a} = \mathbf{d} \tag{8}$$

where C is a $k \times 6$ matrix with rows $\mathbf{c}_1 \ldots \mathbf{c}_k$, and $\mathbf{d}$ is a vector with elements $d_1 \ldots d_k$. Note that (8) should be solved in the least squares sense when there are more than six independent equations. Specifically, we solve the system

$$C^TC\mathbf{a} = C^T\mathbf{d}. \tag{9}$$

This equation predicts the match for any point, even an underconstrained point, providing that an affine transformation is uniquely determined by existing constraint lines in the image.

5 Incorporating Specially Matched Points

In some matching problems, the exact point-to-point correspondence may be known for some special contour points, such as corners, terminators, high curvature points, inflection points, and isolated points. To incorporate the influence of a special point match between $\mathbf{p}_s$ and $\mathbf{p}'_s$, we minimize the distance,

$$|A\mathbf{p}_s + \mathbf{t} - \mathbf{p}'_s| = 0. \tag{10}$$

The two constraint equations for each special point $\mathbf{p}_{s_i}$ are therefore given by

$$\begin{bmatrix} \mathbf{s}_{i1} \\ \mathbf{s}_{i2} \end{bmatrix}\mathbf{a} = \mathbf{p}'_{s_i} \tag{11}$$

where

$$\begin{aligned} \mathbf{s}_{i1} &= [\ x_{s_i} \quad y_{s_i} \quad 0 \quad 0 \quad 1 \quad 0\ ] \\ \mathbf{s}_{i2} &= [\ 0 \quad 0 \quad x_{s_i} \quad y_{s_i} \quad 0 \quad 1\ ]. \end{aligned} \tag{12}$$

Note that these two equations are equivalent to two perpendicular constraint lines that intersect at $\mathbf{p}'_{s_i}$. Again, for a system of q special points, we have

$$S\mathbf{a} = \mathbf{g} \tag{13}$$

where S is a $2q \times 6$ matrix consisting of rows $\mathbf{s}_{11}, \mathbf{s}_{12} \ldots \mathbf{s}_{q1}, \mathbf{s}_{q2}$, and $\mathbf{g}$ is the length $2q$ concatenation of $\mathbf{p}'_{s1} \ldots \mathbf{p}'_{sq}$. Finally, finding the best affine transformation amounts to solving the least squares relation

$$S^TS\mathbf{a} = S^T\mathbf{g}. \tag{14}$$

Since both C^TC and S^TS are positive semidefinite, we can combine the two minimization equations, (9) and (14), and find $\mathbf{a}$ by solving

$$\left((1-\alpha)C^TC + \alpha S^TS\right)\mathbf{a} = (1-\alpha)C^T\mathbf{d} + \alpha S^T\mathbf{g} \tag{15}$$

where α, a number between 0 and 1, is the accuracy of special point matches relative to the accuracy of contour point constraint lines. Again, solving this system of equations for $\mathbf{a}$ determines the best global affine transformation about the global origin, $\mathbf{p}_o$.

6 Matching Locally Planar Contours

Since contours are generally *perspective* projections of *non-planar*, *non-rigid* objects, (15) accurately describes matches only locally. An attractive way to locally enforce the affine assumption involves constructing a local coordinate system at each contour point, $\mathbf{p}_j$. First, the location of each point, $\mathbf{p}_i$, is measured with respect to the local origin, $\mathbf{p}_j$, instead of the global origin, $\mathbf{p}_o$. To make the calculations local we also weigh constraints for each point, $\mathbf{p}_i$, by some locality measure, ω_i. The more local $\mathbf{p}_i$ to $\mathbf{p}_j$, the larger this weight.

Employing the method of weighted least squares to incorporate this weighting scheme into (15), we find the local affine transformation, and hence the match, for each contour point, $\mathbf{p}_j$, by satisfying a system of equations

$$R\mathbf{a} = \mathbf{l} \tag{16}$$

where

$$R = (1-\alpha)C^T W_c^T W_c C + \alpha S^T W_s^T W_s S \tag{17}$$

$$\mathbf{l} = (1-\alpha)C^T W_c^T W_c \mathbf{d} + \alpha S^T W_s^T W_s \mathbf{g}. \tag{18}$$

The diagonal matrices W_c and W_s establish the local *neighborhood* at $\mathbf{p}_j$ and are given by

$$\begin{aligned} W_c &= \mathrm{diag}(\omega_{c1} \ldots \omega_{ck}) \\ W_s &= \mathrm{diag}(\omega_{s1}, \omega_{s1}, \omega_{s2}, \omega_{s2}, \ldots, \omega_{sq}, \omega_{sq}) \end{aligned} \tag{19}$$

where $c1 \ldots ck$ and $s1 \ldots sq$ are the indices for contour and special points respectively. The 6×6 matrix, R, and the six-dimensional vector, $\mathbf{l}$, can both be written explicitly in terms of point locations, normals, and weights by expanding the matrix definitions. Therefore, the elements of these matrices can easily be calculated in parallel for each local neighborhood using simple adders.

7 Oriented Elliptical Neighborhoods

The weight for each point determines the relative influence of its constraints upon the solution. The set of all point weights therefore determines the extent and shape of the local neighborhood. We determine the weights according to several neighborhood criteria. First, the neighborhood integration should monotonically decrease with distance from the local origin, since distant points are less likely to be coplanar with $\mathbf{p}_j$. Second, the neighborhood should be smooth so that matching solutions vary continuously along contours. (Note, however, that this will not mandate the *smoothest* solution along the contour.) Finally, the neighborhood should be maximally elongated and oriented along the contour in order to integrate information primarily along connected proximal contours without serial contour tracing.

With these criteria in mind, we suggest that the weight for $\mathbf{p}_i$ actually be the product of two Gaussian weights:

$$\omega_i = \gamma_i \mu_i \tag{20}$$

as illustrated in Figure 3. The first weight, γ_i, is given by

$$\gamma_i = \exp\left(-|\mathbf{p}_i|^2 / 2\sigma_\gamma^2\right). \tag{21}$$

The set of all γ's constructs a circularly symmetric Gaussian neighborhood about the local origin. The standard deviation of this Gaussian, σ_γ, is the effective size of this neighborhood. The second weight, μ_i, is given by

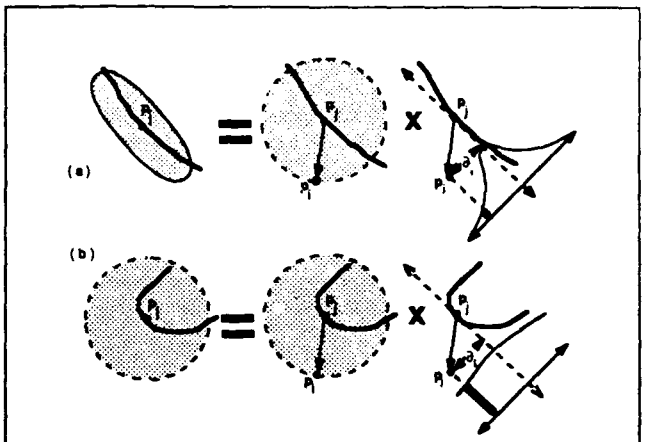


Figure 3: *(a) Integration along nearly linear contour segments is primarily along the contour, while (b) integration over symmetric sections of the contour is circular. Oriented local neighborhoods are determined by the product of $G(|\mathbf{p}_i|)$, a Gaussian of the distance from the local origin ($\mathbf{p}_j$), and $G(\delta_i)$, a Gaussian of the distance from the axis of local orientation passing through $\mathbf{p}_j$. The width of the latter Gaussian is determined by the strength of the local orientation.*

$$\mu_i = \exp\left(-\delta_i^2 / 2\sigma_\mu^2\right) \tag{22}$$

where δ_i is the perpendicular distance from $\mathbf{p}_i$ to axis of local orientation passing through $\mathbf{p}_j$. It modulates the circularly symmetric Gaussian and orients the resulting elliptical neighborhood to integrate information primarily over connected contours. The width, σ_μ, of this Gaussian ranges between 0 and ∞, corresponding to maximal local orientation and circular symmetry, respectively. Hence, the stronger the local orientational preference of the contour, the higher the aspect ratio of the elliptical neighborhood, and the larger the relative influence of points that lie closer to the local axis of orientaion.

We now determine the parameters δ_i and σ_μ, keeping in mind that the the oriented neighborhood must be narrow for linear contours and wide for circularly symmetric contours. To capture this notion, we propose that the major and minor axes of the elliptical neighborhood be respectively aligned with the axes of minimum and maximum local inertia and proportional to the maximum and minimum local second moments. These quantities can be derived through principal component analysis of the inertia matrix

$$J = \begin{bmatrix} J_{xx} & J_{xy} \\ J_{xy} & J_{yy} \end{bmatrix} \tag{23}$$

where

$$\begin{aligned} J_{xx} &= \textstyle\sum_{i=1}^{k} \gamma_{c_i}^2 x_{c_i}^2 \\ J_{xy} &= \textstyle\sum_{i=1}^{k} \gamma_{c_i}^2 x_{c_i} y_{c_i} \\ J_{yy} &= \textstyle\sum_{i=1}^{k} \gamma_{c_i}^2 y_{c_i}^2 \end{aligned} \tag{24}$$

are the second local moments about the local y, $-xy$, and x axes respectively. First, the second moment extremes are simply the eigenvalues of J,

$$\lambda_{max} = \frac{J_{xx} + J_{yy} + \rho}{2}, \qquad \lambda_{min} = \frac{J_{xx} + J_{yy} - \rho}{2} \quad (25)$$

where

$$\rho = \sqrt{(J_{xx} - J_{yy})^2 + (2J_{xy})^2}. \quad (26)$$

Since the major axis is simply $\sigma_\gamma = \beta\lambda_{max}$, the minor axis is $\beta\lambda_{min} = \sigma_\gamma \frac{\lambda_{min}}{\lambda_{max}}$ (β is some proportionality constant). This desired minor axis is the effective width of the modulated Gaussian in the direction perpendicular to the axis of orientation, or

$$\sigma_\gamma \frac{\lambda_{min}}{\lambda_{max}} = (\sigma_\gamma^{-2} + \sigma_\mu^{-2})^{-\frac{1}{2}}. \quad (27)$$

Therefore,

$$\sigma_\mu = \sigma_\gamma \frac{\lambda_{min}/\lambda_{max}}{\sqrt{1 - (\frac{\lambda_{min}}{\lambda_{max}})^2}}. \quad (28)$$

Next, δ_i can be found by dotting $\mathbf{p}_i$ with the unit normal to the axis of local orientation, which is simply the unit eigenvector of J corresponding to the eigenvalue λ_{max}:

$$\hat{\Theta}_\perp = \frac{\begin{bmatrix} \sqrt{\rho + \text{sign}(J_{xy})(J_{yy} - J_{xx})} \\ -\text{sign}(J_{xy})\sqrt{\rho + \text{sign}(J_{xy})(J_{xx} - J_{yy})} \end{bmatrix}}{\sqrt{2\rho}}. \quad (29)$$

Hence,

$$\delta_i = \hat{\Theta}_\perp^T \mathbf{p}_i \quad (30)$$

(see [16] for a more complete analysis of using the principal component method to find the orientational preference of binary images).

Note that, although there may be problems with finding the orientation of a contour that is surrounded by several other contours, surrounding contours should not in general affect the final solution long as we have chosen a sufficiently small neighborhood in which to carry out the calculations. This issue is addressed next.

8 Choosing From Multiple Neighborhood Scales

If the local neighborhood is too small, the affine transformation may not be determined uniquely and is therefore easily degraded by noise. On the other hand, if the neighborhood is too large, the local planarity assumption becomes less accurate, and we also run the risk of including within the same neighborhood independently matched contours. We determine the optimal scale of each local neighborhood by simultaneously examining the constraints for several spatial scales (several values of σ_γ) computed in parallel. Specifically, we select the solution at the smallest spatial scale for which the condition number of R is lower than some number, κ_{max}. In serial simulation this criterion allows us to try smaller scales, which require less integration time, before larger ones. Furthermore, it does not require that we actually solve (16) at each scale. Note that κ_{max} is directly related to the amount of expected noise in the constraint information. One must set this parameter high when available constraints are known to be accurate, and low when high levels of noise exist. Later in the paper we heuristically determine this parameter for simulation purposes.

9 Minimal Solutions for Ambiguous Cases

Despite selection of the optimal neighborhood, the solution to (16) can be underdetermined or unstable, even in the largest neighborhood. This situation occurs when there exists more than one affine mapping between two contours. Consider, for example, a linear contour where the constraint lines for all points are identical. In this case any combination of scaling or shearing along the constraint line, in addition to any translation taking the line to the constraint line, satisfies the constraints. Another example is the matching of concentric circles. If at each point along the first circle the constraint line is parallel to the tangent to the circle at that point, then any amount of rotation, coupled with an appropriate scaling, is possible (see examples in Figure 4).

For these singular cases, we select the *minimal solution*, which minimizes

$$\Delta = \sum_{i=1}^{k} (\omega_{fi} |A\mathbf{p}_{f_i} + \mathbf{t} - (\mathbf{p}_{f_i} + \mathbf{t}_{min})|)^2 \quad (31)$$

where n is the number of points (not just contour points) in the neighborhood. Δ summarizes over the entire neighborhood the squared deviation between predicted point matches and $\mathbf{t}_{min}$, the smallest pure translation that satisfies neighborhood constraints in the least squares sense.

There are a few reasons for minimizing Δ. First, it yields intuitive results because the default is simply a purely translational matching that is closest to the normal component of the match, as illustrated in Figure 4. Any amount of rotation of the circle, or any amount of tangential translation of the line, results in a set of match vectors that deviate more from their normal components. Second, minimising Δ uniquely (stably) determines the match, regardless of the constraint information. Even for collinear points with parallel constraint lines (which can be matched by many purely translational transformations), the average neighborhood normal component is itself the only matching solution that minimizes Δ. Furthermore, Δ is robust to noise because it integrates information over the entire local neighborhood, as opposed to directly minimising the deviation between the solution and the normal component, which is sensitive to noise in the constraint line. Third, minimising Δ preserves continuous matches along contours. As before, calculation over Gaussian local neighborhoods guarantees continuous variation of Δ. Finally, a closed form solution for minimising Δ is efficiently calculated. The optimal solution is selected at each point independently of the final solutions at other points, as opposed to global selection methods which select the set of contour matches to optimize an overall measure, such as smoothness along the contour.

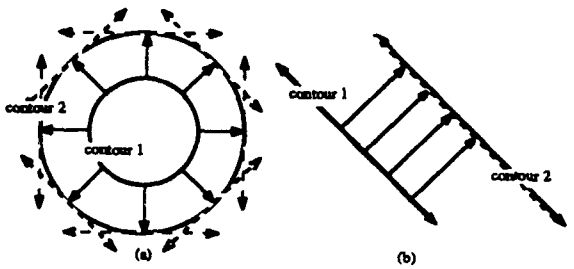


Figure 4: *Minimal solutions for ambiguous transformations are intuitive: (a) concentric circles could be matched with any amount of rotation, but pure expansion should be the default; (b) parallel lines could be matched using a translation in any direction within 90 degrees of the normal component, but the default should be the normal component.*

Given $\mathbf{t}_{min}$, minimising Δ is identical to finding the affine transformation that matches special points. Therefore, to minimize Δ we solve a system of linear equations, similar to those of (13) with $\mathbf{p}_{f_i}' = \mathbf{p}_{f_i} + \mathbf{t}_{min}$, given by

$$F^T W_f^T W_f F \mathbf{a} = F^T W_f^T W_f \mathbf{v} \tag{32}$$

Here F is a $2n \times 6$ dimensional matrix with rows $\mathbf{f}_{11}, \mathbf{f}_{12} \ldots \mathbf{f}_{n1}, \mathbf{f}_{n2}$, where

$$\begin{aligned} \mathbf{f}_{i1} &= \begin{bmatrix} x_{f_i} & y_{f_i} & 0 & 0 & 1 & 0 \end{bmatrix} \\ \mathbf{f}_{i2} &= \begin{bmatrix} 0 & 0 & x_{f_i} & y_{f_i} & 0 & 1 \end{bmatrix}, \end{aligned} \tag{33}$$

$\mathbf{v}$ is a length $2n$ concatenation of $\mathbf{v}_1 \ldots \mathbf{v}_n$, where

$$\mathbf{v}_i = \mathbf{p}_{f_i} + \mathbf{t}_{min}, \tag{34}$$

and W_f is the diagonal matrix of weights for neighborhood points given by

$$W_f = \mathrm{diag}(\omega_{f1}, \omega_{f1}, \omega_{f2}, \omega_{f2}, \ldots, \omega_{fn}, \omega_{fn}). \tag{35}$$

To minimize Δ subject to the matching constraints we solve

$$\min_{\mathbf{a}} (\mathbf{a}^T Q \mathbf{a} - 2\mathbf{a}^T \mathbf{h}) \text{ subject to } R\mathbf{a} = \mathbf{l} \tag{36}$$

where

$$Q = F^T W_f^T W_f F, \qquad \mathbf{h} = F^T W_f^T W_f \mathbf{v}. \tag{37}$$

Once again, $\mathbf{t}_{min}$ is the smallest least squares neighborhood pure translation. In the next section we simplify the existing constraints in order to directly compute this default, and eventually the match vector itself, later in the paper.

10 Simplifications Arising from an Implicit Formulation

Before deriving closed form solutions for $\mathbf{t}_{min}$ and $\mathbf{a}$, it is useful to consider the final match given by the local affine transformation applied to the local origin itself,

$$\mathbf{p}'_j = A\mathbf{p}_j + \mathbf{t} = A \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \mathbf{t} = \mathbf{t}. \tag{38}$$

Since the match itself depends only on the translational component, we need only implicitly solve for the rotation, scaling, and shearing components and thereby directly compute the match vector. In fact, an implicit formulation should be more efficient and stable. Increased efficiency follows from inverting 2×2 and 4×4 submatrices of the 6×6 R matrix instead of inverting R itself. Increased stability follows from analysis of the elements of R: since the coefficients of $\mathbf{t}$ do not depend on the size of the local neighborhood, while the coefficients of A grow as the square of the local neighborhood, uniform random noise affects these elements differently. By separating the translational components, this undesirable property should disappear.

First, let (16) be rewritten as

$$\begin{bmatrix} R_r & R_c \\ R_c^T & R_t \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \mathbf{t} \end{bmatrix} = \begin{bmatrix} \mathbf{l}_r \\ \mathbf{l}_t \end{bmatrix} \tag{39}$$

where R_t is a 2×2 matrix relating the translational affine coefficients, $\mathbf{t}$, to the 2-dimensional vector, $\mathbf{l}_t$, R_r is a 4×4 matrix relating the rotational, shearing, and scaling affine coefficients, $\mathbf{r}$, to the 4-dimensional vector, $\mathbf{l}_r$, and R_c is a 4×2 coupling matrix. Likewise, Q and $\mathbf{h}$ can be represented by

$$Q = \begin{bmatrix} Q_r & Q_c \\ Q_c^T & Q_t \end{bmatrix}, \qquad \mathbf{h} = \begin{bmatrix} \mathbf{h}_r \\ \mathbf{h}_t \end{bmatrix}. \tag{40}$$

At this point, it is advantageous to expand and simplify several of these submatrices. Expanding, we obtain

$$Q_r = \begin{bmatrix} Q_e & 0 \\ 0 & Q_e \end{bmatrix} \tag{41}$$

$$Q_t = \left(\sum_{i=1}^{n} \omega_{fi}^2 \right) I \tag{42}$$

$$Q_c = 0 \tag{43}$$

$$\mathbf{h}_t = \sum_{i=1}^{n} \omega_{fi}^2 \begin{bmatrix} x_{f_i} + \mathbf{t}_{min_x} \\ y_{f_i} + \mathbf{t}_{min_y} \end{bmatrix} = \left(\sum_{i=1}^{n} \omega_{fi}^2 \right) \mathbf{t}_{min}. \tag{44}$$

(taking advantage of the fact that $\sum_{i=1}^{n} \omega_{fi}^2 x_{f_i} = \sum_{i=1}^{n} \omega_{fi}^2 y_{f_i} = 0$ for elliptical neighborhoods due to odd symmetry), where

$$Q_e = \sum_{i=1}^{n} \omega_{fi}^2 \begin{bmatrix} x_{f_i}^2 & x_{f_i} y_{f_i} \\ x_{f_i} y_{f_i} & y_{f_i}^2 \end{bmatrix} \tag{45}$$

and I is the 2×2 identity matrix. Note that the elements of Q_e are second moments taken within the elliptical neighborhood. With this in mind, these second moments should be roughly proportional to the second moments, J_{xx}, J_{xy}, and J_{yy}, that are used to derive this neighborhood (see (24)). Therefore, a reasonable estimation is

$$Q_e \approx J. \tag{46}$$

11 Determining The Default Translation

Using (39), finding $\mathbf{t}_{min}$ for the neighborhood is much like finding the best local affine transformation with the rotational, shearing, and scaling components set to zero (*i.e.* $A = I$, the 2×2 identity matrix), or

$$\mathbf{r} = \mathbf{r}_{min} = [\ 1 \quad 0 \quad 0 \quad 1\ ]^T. \tag{47}$$

Thus, we simply solve a system of equations

$$R_t \mathbf{t}_{min} = (\mathbf{l}_t - R_c^T \mathbf{r}_{min}). \tag{48}$$

Since the determination of a purely translational matching merely requires the intersection of two constraint lines, this system of equations is underconstrained only when there are no special point matches and the constraint lines are parallel. In this case, we choose the solution closest to the average normal component by finding, using the general pseudoinverse (denoted by $+$), the smallest translation satisfying the constraints. Hence,

$$\mathbf{t}_{min} = R_t^+ (\mathbf{l}_t - R_c^T \mathbf{r}_{min}). \tag{49}$$

The pseudoinverse formulation used for this and all subsequent calculations is presented in Appendix B. It differs from the conventionally used form in several ways. First, to promote stability, the SVD threshold is chosen such that the maximum allowed condition number, κ_{max} (introduced previously), mandates the absolute minimum eigenvalue for a given matrix. Second, values below this threshold are not immediately deemed singular, but rather continuously default to zero as they drop below the threshold. Without this latter modification, matching solutions along contours may vary discontinuously, defeating the smoothing properties of the Gaussian neighborhoods and producing rather non-intuitive results.

12 Direct Match Determination

Making use of the separation of affine components and the simplifications made possible by this separation, (36) is equivalent to

$$\min_{\mathbf{t},\mathbf{r},\mathbf{\Gamma}_t,\mathbf{\Gamma}_r} \begin{pmatrix} \mathbf{r}^T(Q_r\mathbf{r} - 2\mathbf{h}_r) \\ \quad + \left(\sum_{i=1}^n \omega_{fi}^2\right) \mathbf{t}^T(\mathbf{t} - 2\mathbf{t}_{min}) \\ \quad + \mathbf{\Gamma}_r^T(R_r\mathbf{r} + R_c\mathbf{t} - \mathbf{l}_r) \\ \quad + \mathbf{\Gamma}_t^T(R_t\mathbf{t} + R_c^T\mathbf{r} - \mathbf{l}_t) \end{pmatrix} \tag{50}$$

where $\mathbf{\Gamma}_t$ and $\mathbf{\Gamma}_r$ are the Lagrange multipliers for the translational and rotational parts of the constraints respectively. Taking the partial derivatives with respect to $\mathbf{\Gamma}_t$, $\mathbf{\Gamma}_r$, $\mathbf{t}$ and $\mathbf{r}$ and setting them to zero respectively leaves

$$R_c^T\mathbf{r} + R_t\mathbf{t} = \mathbf{l}_t \tag{51}$$

$$R_r\mathbf{r} + R_c\mathbf{t} = \mathbf{l}_r \tag{52}$$

$$\mathbf{t} = \frac{\mathbf{x}_t}{\sum_{i=1}^n \omega_{fi}^2} + \mathbf{t}_{min} \tag{53}$$

$$\mathbf{r} = Q_r^{-1}\mathbf{x}_r + \mathbf{r}_{min} \tag{54}$$

where

$$\mathbf{x}_t = -\frac{1}{2}(R_c^T\mathbf{\Gamma}_r + R_t^T\mathbf{\Gamma}_t), \qquad \mathbf{x}_r = -\frac{1}{2}(R_r^T\mathbf{\Gamma}_r + R_c\mathbf{\Gamma}_t) \tag{55}$$

since $Q_r^{-1}\mathbf{h}_r = \mathbf{r}_{min}$, the default for rotational, shearing, and scaling components.

Finally, substitution of (53) and (54) into (51) and (52) then allows us to solve for $\mathbf{x}_t$ and derive

$$\mathbf{t} = (R_t - R_c^T R_r^{\#} R_c)^+ \begin{pmatrix} \mathbf{l}_t - R_c^T R_r^{\#}\mathbf{l}_r \\ \quad - (R_c^T R_r^{\#} R_c - R_t)\mathbf{t}_{min} \\ \quad - (R_c^T R_r^{\#} R_r - R_c^T)\mathbf{r}_{min} \end{pmatrix} + \mathbf{t}_{min} \tag{56}$$

where

$$R_r^{\#} = Q_r^{-1}(R_r Q_r^{-1})^+ = \begin{bmatrix} J^{-1}|J| & 0 \\ 0 & J^{-1}|J| \end{bmatrix} \left(R_r \begin{bmatrix} J^{-1}|J| & 0 \\ 0 & J^{-1}|J| \end{bmatrix} \right)^+ \tag{57}$$

and

$$J^{-1}|J| = \begin{bmatrix} J_{yy} & -J_{xy} \\ -J_{xy} & J_{xx} \end{bmatrix}. \tag{58}$$

Equation (56) not only minimizes the overall deviation between neighborhood matches and the best pure translation, but it also guarantees a unique, stable match regardless of the constraints or the neighborhood. When R_r and Q_r are non-singular, then $R_r^{\#} = R_r^{-1}$, and the default, $\mathbf{r}_{min}$, factors out of the solution. Otherwise, one must use the pseudoinverse. The latter case occurs when the neighborhood contour points are collinear, since at least three non-collinear point matches are required to uniquely define an affine transformation. When this situation occurs, Q_r affects the solution by altering the eigenvalues or R_r. It essentially transforms the Euclidean affine parameter space to one in which finding the closest affine solution vector to the purely translational affine transformation,

$$\mathbf{a}_{min} = \begin{bmatrix} \mathbf{r}_{min} \\ \mathbf{t}_{min} \end{bmatrix}. \tag{59}$$

yields an affine transformation that minimizes Δ.

13 Implementation and Results

The matching scheme was implemented in C on a Sun workstation. Five spatial scales were employed. The sizes, in terms of the variance of the local Gaussian neighborhood, σ_γ, were 4, 8, 16, 32, and 64 pixels. These sizes were chosen so that the largest neighborhood was on the order of the size of the examples, thereby ensuring that the largest local neighborhood could roughly include the constraints of the entire example. Input consisted of two 256×256 pixel binary contour images. Some were synthetically produced, while others were extracted zero crossings of smoothed and differentiated natural imagery [22]. Though the method should be able to deal with any set of contour images with partially constrained matches, the two images chosen for each simulation were relatively aligned to reflect the fact that constraint lines are most easily derived from mildly differing

imagery (*i.e.* time sampled imagery for short-range motion, or roughly aligned objects for object recognition). The "correct" matches between each pair of images were known for synthetic imagery and hand-picked for natural imagery. The constraint line for each contour point was determined by finding the local orientation of the contour, then taking the line with the same orientation that passed through the "correct" match. In the context of visual motion, these constraint lines roughly mimicked a local measurement of normal velocity and allowed comparisons to be made with the results of previous methods for recovering optical flow.

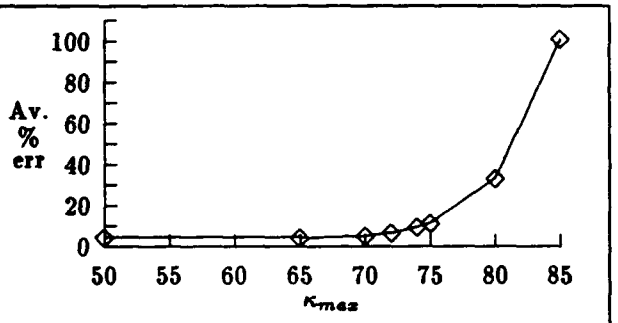


Figure 5: *Average Percent Matching Error vs.* κ_{max}.

To assess the affect of noise upon the matching process, 10% random noise was added to the components of the normals describing the constraint lines and special point matches. Assuming this equal noise distribution, a value of 0.5 was used for α, the accuracy of special points relative to the accuracy of constraint lines. Furthermore, the parameter κ_{max}, used both for selecting the size of the neighborhoods and for setting the SVD threshold in pseudoinverse operations, was experimentally determined by applying the scheme to the worst case (most ambiguous) matching problem, the matching of parallel lines (see example 1b in appendix A). The average percentage deviation of matches with respect to the actual normal components of the match (the expected minimal solution) was determined for several values of κ_{max}. Using the results, shown in Figure 5, the tradeoff between stability (low κ_{max}) and accuracy (high κ_{max}) was balanced by choosing $\kappa_{max} = 74.0$, the highest value for which the average percentage error did not exceed the percentage noise. This value was used for all subsequent examples, the results of which are shown in Appendix A. Note that this choice seems particularly appropriate considering the large increase in matching error as κ_{max} increases beyond this value.

14 Discussion of Results

As demonstrated by the results presented in Appendix A, the proposed contour matching scheme robustly recovers the correspondences of both synthetic and natural contour imagery. First, unlike most other methods for matching contours, this scheme yields intuitive solutions for ambiguous cases, such as parallel lines. A second attribute of the scheme concerns the usefulness of specially matched points. While special points obviously aid the matching process of the ambiguously matched parallel lines, the scheme did not exploit any special points for subsequent examples and nevertheless recovered the actual correspondences. This observation indicates that, while special points are especially important for the matching of ambiguous examples, they are no more powerful than redundant constraint lines when used in conjunction with an unambiguous example.

A third observation is that analysis at several neighborhood scales greatly enhances the scheme's ability to recover exact matches, particularly those collective matches not modeled precisely by a global affine transformation. For example, the three-dimensional distortions used to generate the space curve did not present a problem to the matching scheme. The attribution of this capability to several neighborhood sizes is not obvious from the results, but instead from simulations that reported the sizes of the selected neighborhoods at every contour point. Matching the views of the face, for example, required small neighborhoods for matching contours with intricate detail (such as the eyes and nose) and large neighborhoods for matching nearly linear contours (such as the occluding boundary between the face and the background). It is not until we examine the matches predicted for the three-dimensional rotated wire frame that we see the limitations of multiple scales. Though the overall matching solution is quite accurate, the matching of overlapping contours undergoing separate motion is highly inaccurate because information has *in this* case been integrated over independent contours.

15 Conclusion and Future Work

A method for determining matches between contours in two images has been proposed, developed, and tested, assuming that constraint lines, each narrowing down the match for a contour point in the first image to a line in the second image, are available for several contour points in the first image. Suggested applications include the determination of optical flow in short-range motion and the matching of aligned contour views in either alignment-based object recognition or long-range motion.

To determine the match for a contour point we find the best affine transformation, in a weighted least squares sense, that satisfies the match constraint lines and specially matched points within an oriented elliptical neighborhood. This neighborhood is established by weighing the constraint equations. The weight for a given point constraint is the modulation of the Gaussian distance to the local origin, which establishes a circularly symmetric local neighborhood, by the Gaussian distance from the axis of local orientation, which attempts to limit the neighborhood to a single contour. The width of the modulating Gaussian is set such that the axes of the elliptical neighborhood are proportional to the local axes of inertia. To determine the width of the circularly symmetric Gaussian, the effective size of the local neighborhood, we consider several sizes simultaneously and choose the smallest one which yields a stable, unique solution. Stability is tested by comparing the condition number of the constraint matrix, R, with the maximum condition num-

ber, κ_{max}, set according to the expected noise level. In the event that this condition number exceeds κ_{max} even at the largest neighborhood, there is more than one possible solution, from which we choose the smallest affine transformation which predicts the set of neighborhood matches which deviate the least from the smallest least squares purely translational matching. Unique determination of both the smallest pure translation and the final match is guaranteed through the use of a modified general pseudoinverse.

Note that the match for each point may be computed in parallel. Furthermore, since the predicted match for a point is simply given by the translational component of the local affine transformation, finding the match involves explicitly solving for only the two translational components of the affine transformation. A closed form solution for the match involves using a continuous version of the pseudoinverse to invert a 2 × 2 matrix and a 4 × 4 matrix, the coefficients of which are weighted summations of local point constraint parameters that may be determined in parallel.

Simulation results show that the scheme performs well for most examples, including noisy synthetic imagery and edges extracted from natural imagery. Minimal solutions obtained when the recovery is ill-posed are intuitive. Incorporation of terminators improves the correspondence significantly for near-ambiguous examples.

Despite the scheme's performance, further research remains. First, the precise conditions under which the local affine transformation is uniquely determined by the constraints should be better understood. Second, simulations should more rigorously test matching capabilities and limitations. Third, the application of the scheme to recovering optical flow, matching features for object recognition, and finding correspondence for long-range motion can be addressed. Finally, extensions can be considered, including the use of depth information in the formation of ellipsoidal neighborhoods that avoid difficulties caused by overlapping, independently matched contours, and the incorporation of prediction mechanisms (such as Kalman filtering techniques) for applications that are aided by the temporal improvement of matching solutions over multiple image frames.

16 Acknoledgements

We thank Eric Grimson and Amnon Shashua for looking over drafts of this paper. This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by National Science Foundation, contract number IRI 8900267, and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124.

A Simulation Results

In each example presented the first contour image is shown in black, and the second contour image is superimposed as dotted contours. In column *I*, vectors describing the constraint lines are shown for sampled points along the first contour, and special match vectors are indicated by small squares at the endpoints. 10% noise has been added to these vectors. The computed and actual match vectors for sampled points are then shown in columns *II* and *III*, allowing qualitative comparison. Quantitative results are also presented. At each sampled point, the relative error is computed by normalizing the distance between predicted and actual matches by the length of the actual match vector. The average relative error, computed over all sampled points, is reported for each example. Note that predicted matches do not necessarily lie exactly upon the second contour, especially for the matching of ambiguous examples. In practice, final matches may be found by taking the point on the second contour that is closest to the predicted match.

Rows 1a and 1b respectively present the matching of synthetic parallel lines with and without terminator matches. In row 2a an orthographic projection of a synthetic wire-frame is matched for purely translational correspondence, and in row 2b the same wire-frame is rotated by 20 degrees about each axis and matched. Row 3 presents the matching of an orthographic projection of an arbitrary synthetic 3D space curve rotated by 10 degrees about each axis, translated, scaled by a factor of 1.2, and stretched by a factor of 1.1 along each axis. Finally, the matching of the edges obtained from a pair of roughly aligned natural views of a pair of scissors (borrowed from [6]), a tank, and a doll face (borrowed from [29]) are respectively presented in rows 4 through 6.

B Modified Pseudoinverse

This appendix describes a modified general pseudoinverse. First, an $n \times m$ matrix A is uniquely diagonalized using singular value decomposition techniques (SVD):

$$A = Q_1 \Lambda Q_2^T \tag{60}$$

where Q_1 and Q_2 are $n \times m$ and $m \times n$ orthonormal matrices respectively, Λ is the $m \times n$ matrix of the singular values of A,

$$\Lambda = \begin{bmatrix} \sqrt{\lambda_1} & & 0 & 0 & \cdots & 0 \\ & \ddots & & \vdots & & \vdots \\ 0 & & \sqrt{\lambda_n} & 0 & \cdots & 0 \end{bmatrix} \tag{61}$$

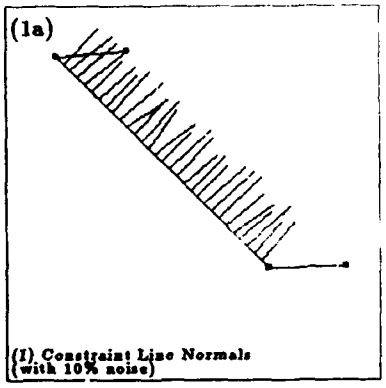
(in this particular case for $n \leq m$) and $\lambda_1 \cdots \lambda_n$ are the eigenvalues of $A^T A$. Given this diagonalization, the modified pseudoinverse is

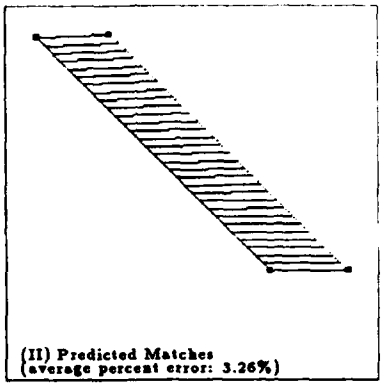
$$A^+ = Q_2 \Lambda^+ Q_1^T \tag{62}$$

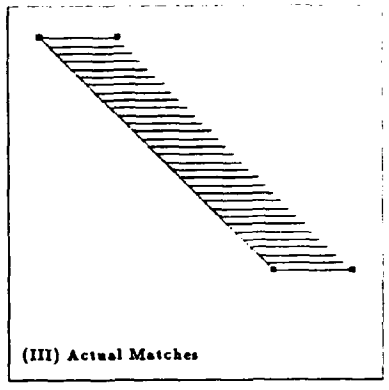
where

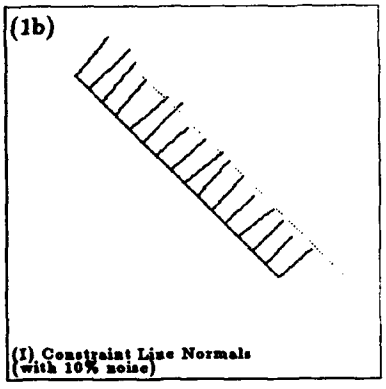
$$\Lambda_{ij}^+ = \begin{cases} 1/\Lambda_{ij} \text{ if } \Lambda_{ij} \geq \sqrt{\frac{\lambda_{max}}{\kappa_{max}}} \\ \Lambda_{ij} \frac{\kappa_{max}}{\lambda_{max}} \text{ otherwise} \end{cases} . \tag{63}$$

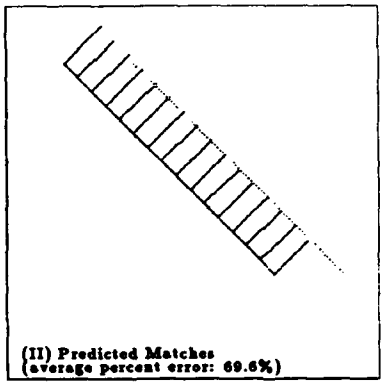
As a first modification, the elements of Λ^+ are continuous functions of the elements of Λ and gradually default to zero. As a second modification, the singular value threshold is chosen for stability by guaranteeing that the effective condition number of $A^T A$, the ratio of the maximum to minimum eigenvalues, is at most κ_{max}. (See [30] for a more detailed presentation of SVD and the general pseudoinverse.)

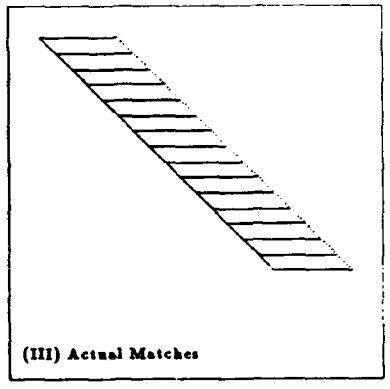

(1a)
(I) Constraint Line Normals
(with 10% noise)

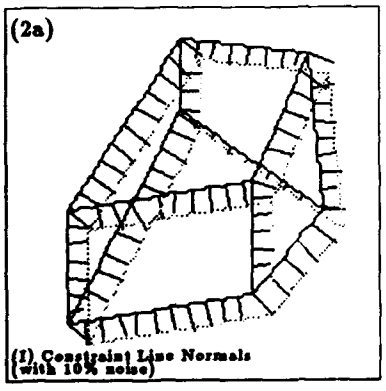

(II) Predicted Matches
(average percent error: 3.26%)

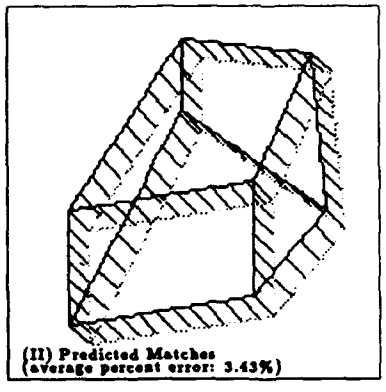

(III) Actual Matches

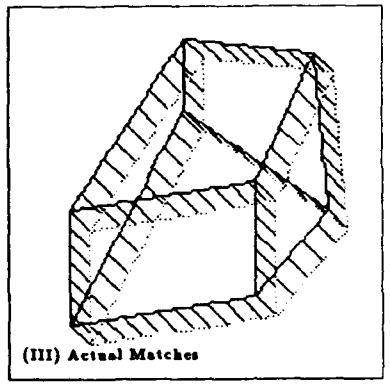

(1b)
(I) Constraint Line Normals
(with 10% noise)

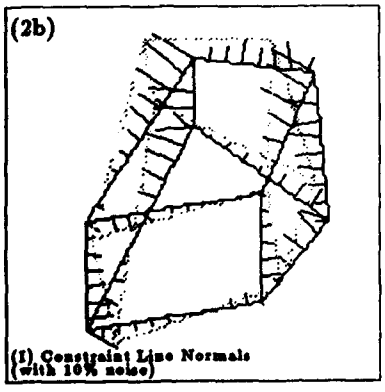

(II) Predicted Matches
(average percent error: 69.6%)

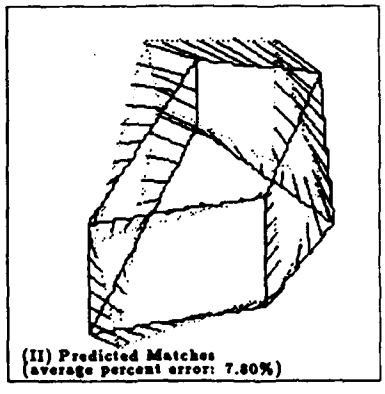

(III) Actual Matches

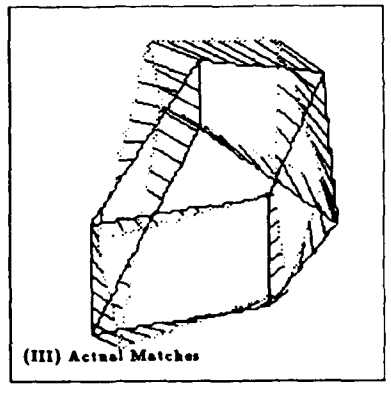

(2a)
(I) Constraint Line Normals
(with 10% noise)

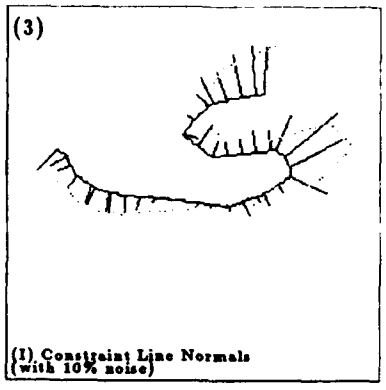

(II) Predicted Matches
(average percent error: 3.43%)

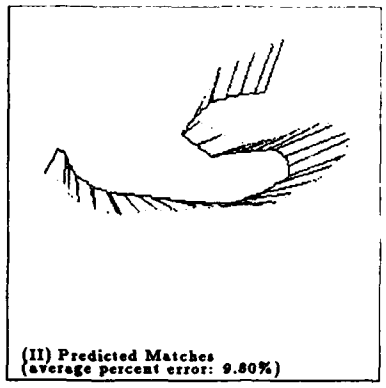

(III) Actual Matches

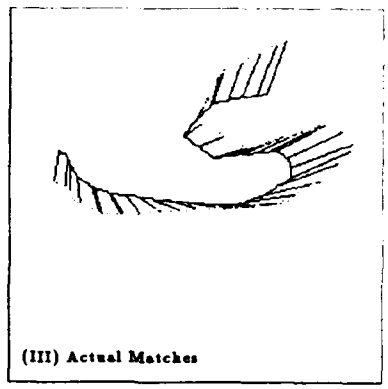

(2b)
(I) Constraint Line Normals
(with 10% noise)

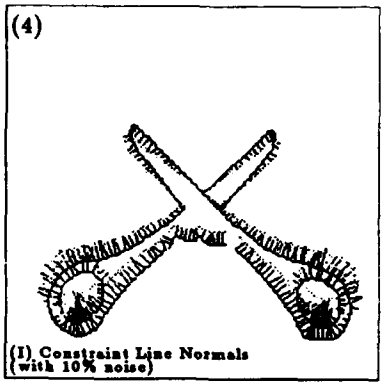

(II) Predicted Matches
(average percent error: 7.80%)

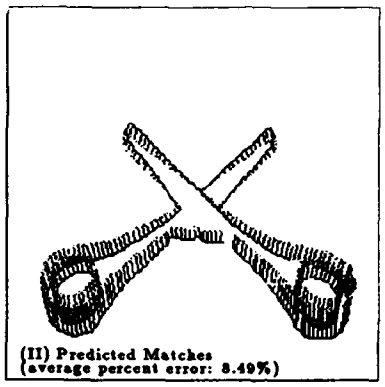

(III) Actual Matches

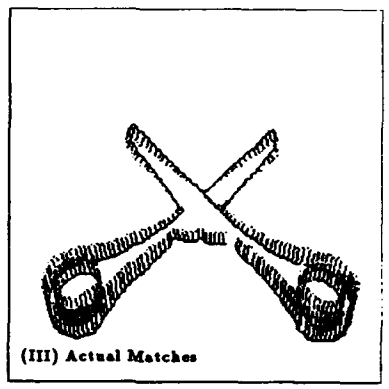

(3)
(I) Constraint Line Normals
(with 10% noise)

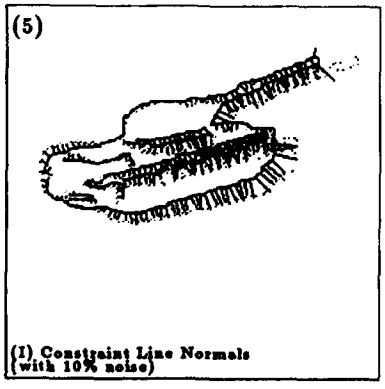

(II) Predicted Matches
(average percent error: 9.80%)

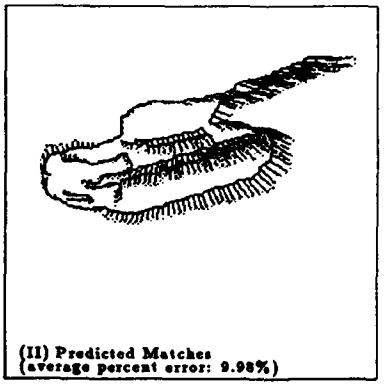

(III) Actual Matches

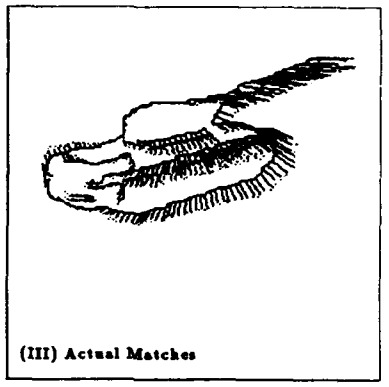

(4)
(I) Constraint Line Normals
(with 10% noise)

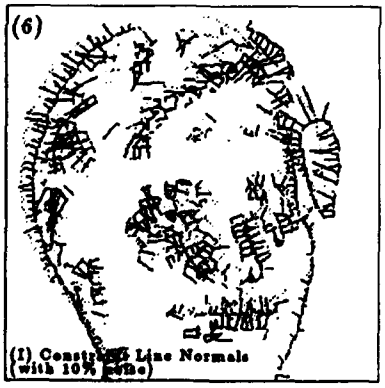

(II) Predicted Matches
(average percent error: 8.49%)


(III) Actual Matches


(5)
(I) Constraint Line Normals
(with 10% noise)


(II) Predicted Matches
(average percent error: 9.98%)


(III) Actual Matches


(6)
(I) Constraint Line Normals
(with 10% noise)


(II) Predicted Matches
(average percent error: 7.58%)


(III) Actual Matches

References

[1] E. H. Adelson and J. A. Movshon. Phenomenal coherence of moving visual patterns. *Nature*, 300:523–525, 1982.

[2] P. Anandan and R. Weiss. Introducing a smoothness constraint in a matching approach for the computation of optical flow fields. In *IEEE Workshop on Computer Vision: Representation and Control*, pages 186–194, Bellaire, MI, October 1985.

[3] K. Arbter. Affine-invarient fourier descriptors. In *COST 13 Workshop*, pages 22–27, Bonas, France, August 1988.

[4] I. A. Bachelder. Contour matching using local affine transformations. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, June 1991.

[5] I. A. Bachelder and S. Ullman. Contour matching using local affine transformations. A.I. Memo 1326 (in press), The Artificial Intelligence Lab., M.I.T., 1991.

[6] R. Basri. *Recognition of 3-D solid objects from 2-D images*. PhD thesis, Weizmann Institute, Rehovot, Isreal, October 1990.

[7] P. J. Burt, J. Bergen, R. Hingorani, S. Peleg, and P. Anandan. Dynamic analysis of image motion for vehicle guidance. In *IEEE International Workshop on Intelligent Motion Control*, pages 75–82, Bogazici University, Istanbul, August 20-22 1990.

[8] P. J. Burt, J. R. Bergen, R. Hingorani, R. Kolczynski, W. A. Lee, A. Leung, J. Lubin, and H. Shvaytser. Object tracking with a moving camera. In *Proceedings of the workshop on visual motion*, pages 2–12, Irvine, CA, March 20-22 1989.

[9] D. Cyganski and J. A. Orr. Applications of tensor theory to object recognition and orientation determination. *IEEE Trans Patt Anal Mach Intell*, PAMI-7(6):662–673, 1985.

[10] C. L. Fennema and W. B. Thompson. Velocity determination in scenes containing several moving objects. *Comput. Graph. Image. Proc.*, 9:301–315, 1979.

[11] J. J. Gibson and E. J. Gibson. Continuous perspective transformations and the perception of rigid motion. *Journal of Experimental Psychology*, 54:129–138, 1957.

[12] E. C. Hildreth. *The Measurement of Visual Motion*. MIT Press, Cambridge, 1984a.

[13] E. C. Hildreth. The computation of the velocity field. *Proc. R. Soc. London B*, 221:189–220, 1984b.

[14] E. C. Hildreth and S. Ullman. The computational study of vision. A.I. Memo 1038, The Artificial Intelligence Lab., M.I.T., April 1988.

[15] J. Hong and X. Tan. The similarity between shapes under affine trasformation. Robotics Res. Rep 133, New York University, December, 1987.

[16] B. K. P. Horn. *Robot Vision*. The MIT Press and McGraw-Hill, Cambridge and NY, 1986.

[17] B. K. P. Horn and B. G. Schunk. Determining optical flow. *Artif. Intell.*, 17:185–203, 1981.

[18] D. P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195–212, 1990.

[19] J. J. Koenderink and A. J. Van Doorn. Local structure of movement parallax of the plane. *Journal of the Optical Society of America*, 66:717–723, 1976.

[20] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson. Affine invariant model-based object recognition. *IEEE Transactions on Robotics and Automation*, 6(5):578–589, 1990.

[21] J. S. Lappin and H. H. Bell. The detection of coherence in moving random dot patterns. *Vision Research*, 16:161–168, 1976.

[22] D. Marr and E. C. Hildreth. Theory of edge detection. *Proc R Soc. London B*, 207:187–217, 1980.

[23] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194:283–287, 1976.

[24] J. E. W. Mayhew and J. P. Frisby. Psychophysical and computational studies towards a theory of human stereopsis. *Artificial Intelligence*, 17:349–385, 1981.

[25] H. H. Nagel. Recent advances in image sequence analysis. In *Proc. Premier Colloque Image - Traitement, Synthese, Technologie et Applications*, pages 545–558, Biarritz, France, May 1984.

[26] H. H. Nagel and W. Enkelmann. Towards the estimation of displacement vector fields by "oriented smoothness" constraints. In *7th Int. Conf. on Pattern Recognition*, pages 6–8, Montreal, Canada, July 1984.

[27] A. J. Pantle and L. Picciano. A multistable display: Evidence for two separate motion systems in human vision. *Science*, 193:500–502, 1976.

[28] G. F. Poggio and T. Poggio. The analysis of stereopsis. *Annual Reviews of Neuroscience*, 7:379–412, 1984.

[29] A. Shashua. Illumination and 3D object recognition. In John E. Moody, Steve Hanson, and Richard Lippmann, editors, *Advances in neural information processing systems 4*. Morgan Kaufmann, 1992 (in press). Proc. NIPS '91, Denver CO.

[30] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Wellesly, MA, 1986.

[31] S. Ullman. Aligning pictorial descriptions: An approach to object recognition. *Cognition*, 32(3):193–254, 1989.

[32] T. Wakahara. On-line cursive script recognition using local affine transformation. *Syst. Comput. Jpn. (USA)*, 20(7):10–19, 1988.

[33] A. M. Waxman and K. Wohn. Contour evolution, neighborhood deformation, and global image flow: Planar surfaces in motion. *International Journal of Robotics Research*, 4:95–108, 1985.

HyperBF Networks for Gender Classification

R. Brunelli[1], T. Poggio[2,1]
[1] Istituto per la Ricerca Scientifica e Tecnologica
I-38050 Povo, Trento, ITALY
[2]Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139, USA

Abstract

A set of geometrical features is extracted automatically from digitised pictures of frontal views of people without facial hair. This compact description is then used to train two competing HyperBF networks to classify according to gender. The results using a database of twenty males and twenty females show an average performance of 79% correct gender classification on images of new faces. Correct classification on vectors corresponding to new face images present in the training set but not used in the training phase rises to 86%. Preliminary experiments to assess human performance on the same set of grey level images give an average result of 90% which, while higher than network performance, suggests that peoples' performance is comparable. Interestingly, the HyperBF technique finds the relative weights of the different features and converges to prototypes of the male and female face that seem to exaggerate their difference, somewhat like caricatures do.

1 Introduction

Faces allow people to establish, among other things, the gender of a person, his (her) age and, to a certain extent, emotions. In the current paper we address gender classification and will show how limited geometrical information accounts for correct sex attribution.

There are two main strategies for face recognition (and for object recognition in general): feature comparison and template matching. The former relies on a set of selected features which must be computed from an available image while the latter directly compares the appearance of a given instance with a reference image by means of a suitable metric. The first strategy, when feasible, works with a compact representation of the objects to be matched which are usually represented by low (as compared to the number of pixels of a template) dimensional vectors. The set of features used for recognition or classification is critical as it must capture the discriminating ones and give to each of them the correct weight.

In some recent work [4] the problem of face recognition and gender classification has been tackled using the internal representation of a compression network as unsupervised feature extractor and a (smaller) classification network taking as input the extracted features. Recent theoretical results [2] show that the internal representation of such a network is closely related to a Karhunen-Loewe expansion (see also [8, 13]) so that the work of Cottrell et al. should probably be considered as classified in the template matching category. In our paper we want to show how limited geometrical information (see Fig. 1 for the set of features) can give reasonable performance and possibly provide some insight into human mechanisms.

2 Gender Classification

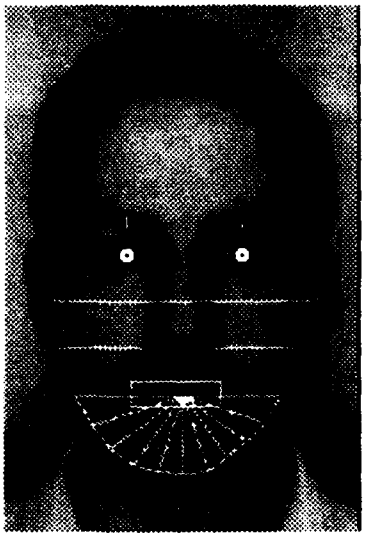
The inspection of a face allows us to establish, usually without much effort, the gender of the person we are looking at. It seems natural to mimic this ability with a computer program. The experiment we did is based on the use of a geometrical feature vector. In fact, the same vector extracted for recognition purposes in a previous paper [3] was used. The only difference is that the face description has been symmetrised (left and right eyebrow and chin information has been averaged) thereby reducing the dimensionality of the vector.

All of the features have been extracted automatically, from images whose rotation and scale was previously normalised (by automatically locating eyes). The paradigm we used is that of learning from examples, where a system learns to discriminate between males and females given a sufficient number of examples. The system we used is based on a classifier called Hyper Basis Function Network (see [10]).

Learning from examples can be regarded, whenever the inputs and output are expressible as numerical vectors, as the reconstruction of an unknown function from sparse data. From this point of view learning is equivalent to functional approximation. Hyper Basis Function Networks are a tool for multivariate function approximation and rest on a solid background of results in this field.

Before describing the networks used for gender classification let us briefly recall the fundamentals of the Hyper Basis Functions Network.

Radial Basis Functions can be regarded as a special case of Regularisation Networks introduced in [10] as a general approximation technique that can be used in



1	pupil to eyebrows separation
2	pupil to nose vertical distance
3	pupil to mouth vertical distance
4	pupil to chin vertical distance
5	eyebrows thickness
6	nose width
7	mouth width
8	bizygomatic breadth
9	bigonial breadth
10-15	six chin radii
16	mouth height

Figure 1: Geometrical features (white) used in the gender classification experiments

problems of learning from examples.

A scalar function can be approximated, given its value on a sparse set of points $\{\bar{x}_i\}$, by an expansion in radial functions:

$$F(\bar{x}) = \sum_{i=1}^{N} c_i G(\| \bar{x} - \bar{x}_i \|) \qquad (1)$$

where $\| \cdot \|$ represents the usual Euclidean norm. The computation of the coefficients c_i rests on the invertibility of matrix $\mathbf{H}_{ij} = G(\| \bar{x}_i - \bar{x}_j \|)$ which has been proved (see Micchelli [9]) for functions such as:

$$G(r) = e^{-(\frac{r}{\sigma})^2} \qquad (2)$$

$$G(r) = (c^2 + r^2)^{\alpha}, \alpha < 1 \qquad (3)$$

It is possible to use fewer radial functions than examples, i.e. data points. The resulting overconstrained system can be solved using a least square approach under the conditions of Michelli's theorem and proves to be useful when many examples are available [10].

Poggio and Girosi [11] have shown that the RBF technique is a special case of the regularization approach to the approximation of multivariate functions. From a more general formulation of the variational problem of regularization they derive the following approximation scheme, instead of equation (1):

$$f^*(\mathbf{x}) = \sum_{\alpha=1}^{n} c_\alpha G(\|\mathbf{x} - \mathbf{t}_\alpha)\|_W^2) + p(\mathbf{x}) \qquad (4)$$

where the parameters $\mathbf{t}_\alpha$, which we call "centers," and the coefficients c_α are unknown, and are in general fewer than the data points ($n \leq N$). The term $p(\mathbf{x})$ is a polynomial that often can be neglected, though it may be useful to keep the constant and linear terms. The norm is a *weighted norm*

$$\|(\mathbf{x} - \mathbf{t}_\alpha)\|_W^2 = (\mathbf{x} - \mathbf{t}_\alpha)^T W^T W (\mathbf{x} - \mathbf{t}_\alpha) \qquad (5)$$

where W is an unknown square matrix and the superscript T indicates the transpose operator. In the simple case of diagonal W the diagonal elements w_i assign a specific weight to each input coordinate, determining in fact the units of measure and the importance of each sensory input. In this formulation the learning stage is used to estimate not only the coefficients of the RBF expansion, but also the metric (problem dependent dimensionality reduction) and the position of the centers (optimal examples selection).

In a classification task, in which the function range is represented by the closed interval $[0, 1]$, the value of the function can be interpreted as a *fuzzy predicate*. If a gaussian function is used the center of expansion is the only point at which the predicate assumes value 1: it can be effectively interpreted as a prototype (note that the use of HyperBF Networks for classification is directly related to Bayes estimation as pointed out in [10]).

Using a geometrical vector as input, gender classification has been attempted by using two competing networks: one for male recognition and one for female recognition (see Fig. 2 for the network structure). The gender to be associated to a given vector is taken to be that corresponding to the network with the greatest response. It is interesting to note how each of the networks is able to create a meaningful prototype of the class it represents. As can be seen in Figure 2 the expansion center, which is a vector with components free to move during the "learning" process, has converged at the end of the training phase to what could be considered a caricature of a (fe)male face. It does not correspond to the average value on the separate subsets: it emphasizes the discriminating features. It is intriguing to speculate that putative cells in IT cortex involved in gender classification may be similar to the units of our model, representing fuzzy (because of the Gaussian) templates such as the ones of Fig. 2. The learning stage is also able to change the metric to account for the different weight and significance of the different features. Of the sixteen features only three are given a noticeable weight: distance of eyebrow from eyes, eyebrows thickness and nose width. These are followed by the vertical position of nose

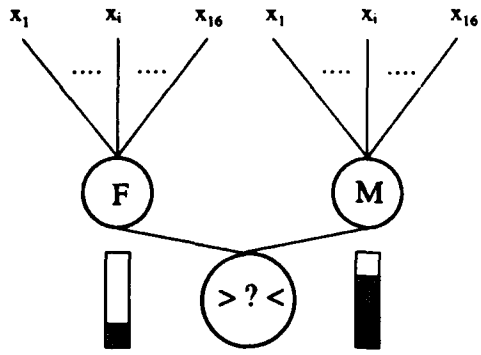


Figure 2: The competing HyperBF Networks used for gender classification

and mouth and the two radii describing the lower chin shape; the remaining features are considered ineffective.

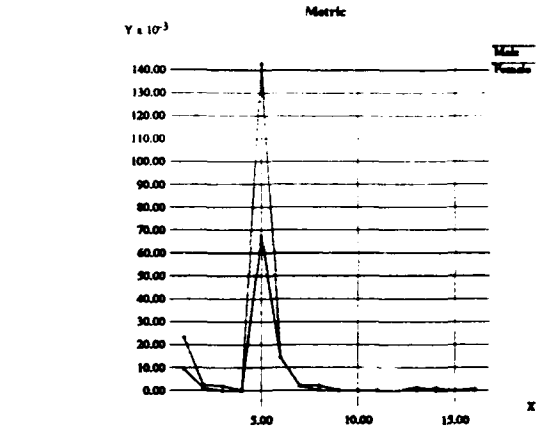
The database used for the classification experiments comprised 168 vectors equally distributed over 21 males and 21 females. Three different performances were measured:

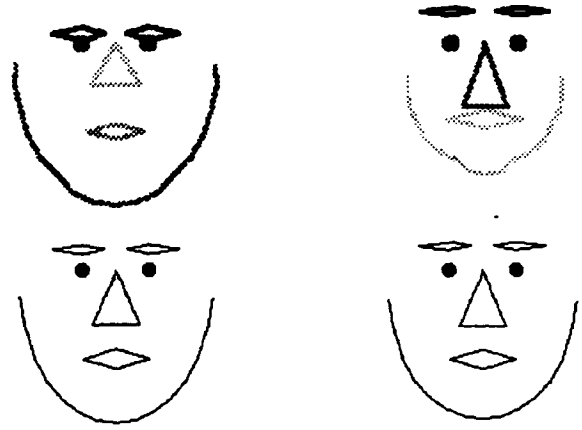
- on the vectors of the training set (90% correct);
- on novel faces of people in the training set (86% correct);
- on faces of people not represented in the training set (79% correct)

The performance on a testing set, having null intersection with the training set, has been estimated with a *leave-one-out* strategy. Having n available examples, training was done on the first $n-1$ data leaving the last one for testing. The data set was then rotated, so that each of the available examples was used in turn as a testing example. The performance was estimated by taking the percentage of correct gender assessment on the resulting tests. The performance obtained is of 79% correct classifications.

Human performance in such classification tasks (as well as recognition) is widely believed to be nearly perfect. To assess the effective ability of people in gender classification we have performed some informal psycophysical experiments using as stimulation pattern a grey level image of the face from which the local average was subtracted (to make the different images as similar as possible). As Figure 3 shows, no hair information was available (residual facial hair was masked out)

The database of stimuli was then presented one image after another on a computer screen and the subject was asked to press M for male and F for female without any time constraint. The results were surprinsing. An average score of 90% correct classification (on 17 subjects





TOP. Feature weights for gender classification as computed by the HyperBF Networks. MIDDLE. The male prototype (left) and the female prototype (right) as synthesized by the HyperBF Networks with movable coefficients, centers and metric. The darker the feature, the more important it is according to the corresponding entries in the diagonal metric W. BOTTOM. Average male face (left) and female face (right).

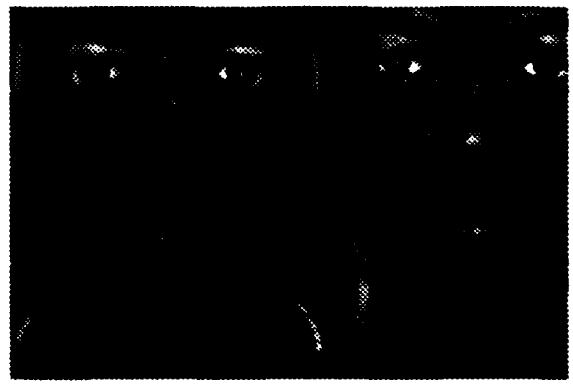


Figure 3: Typical stimuli used in the experiments of human gender classification

some of which familiar with a large subset of the people represented in the database). Classification performance was not impaired by the lack of familiarity with the database people. Informal chat with some of the subjects revealed that, at least consciously, eyebrow information was considered to be the most discriminating.

Note that no hair information has been used, both in our human and our computer gender classification experiments. This must be considered if these results are to be compared with other experiments reported in literature (see for example [6], where images included limited hair information).

3 Conclusion

Gender classification has been attempted using two competing HyperBF networks trained on a geometrical description of (fe)males faces. The resulting performance was of 79% correct classification (averaged on males and females) and must be confronted to a human performance of 90%. Analysis of the *internal representation* of the HyperBF networks shows that the networks have been able to effectively prototype (fe)male faces and that classification was achieved using a subset of the available features, similar to human strategies of gender classification.

References

[1] R. J. Baron. Mechanisms of human facial recognition. *International Journal of Man Machine Studies*, 15:137–178, 1981.

[2] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4/5):291–294, 1988.

[3] R. Brunelli and T. Poggio. Face recognition: Features versus templates. Technical Report 9110-04, I.R.S.T, 1991.

[4] G. Cottrell and M. Fleming. Face recognition using unsupervised feature extraction. In *Proceedings of the International Neural Network Conference*, 1990.

[5] I. Craw, H. Ellis, and J.R. Lishman. Automatic extraction of face features. *Pattern Recognition Letters*, 5:183–187, Feb. 87.

[6] B. A. Golomb, D. T. Lawrence, and T. J. Sejnowski. Sexnet: A neural network identifies sex from human faces. In *Advances in Neural Information Processing Systems 3*, pages 572–577, 1991.

[7] T. Kanade. Picture processing by computer complex and recognition of human faces. Technical report, Kyoto University, Dept. of Information Science, 1973.

[8] M. Kirby and L. Sirovich. Application of the Karhunen-Loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103–108, 1990.

[9] C. A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constr. Approx*, 2:11–22, 1986.

[10] T. Poggio and F. Girosi. A theory of networks for approximation and learning. Technical Report A.I. Memo No. 1140, Massachusetts Institute of Technology, 1989.

[11] T. Poggio and F. Girosi. Networks for approximation and learning. In *Proc. of the IEEE, Vol. 78*, pages 1481–1497, 1990.

[12] J. Sergent. Structural processing of faces. In A.W. Young and H.D. Ellis, editors, *Handbook of Research on Face Processing*. North-Holland, Amsterdam, 1989.

[13] M. Turk and A. Pentland. Eigenfaces for recognition. Technical Report 154, MIT Media Lab Vision and Modeling Group, 1990.

Sensitivity of Feature Configuration in Viewpoint Determination*

Karen T. Sutherland
Computer Science Department
University of Minnesota
Minneapolis, MN 55455

Abstract

Navigation using maps requires establishing a match between locations in the environment and locations on a map (localization). Viewpoint can be determined by using the visual angles between three features. However, error in estimate of visual angles produces error in localization. Depending on feature configuration, the same error in visual angle estimate can result in very different errors in localization. We show how "areas of uncertainty" change as configuration shape and relation to observer are modified and describe properties of configurations which can be used to predict error in localization. Two basic conditions have been identified which affect area size and shape.

1 Introduction

Visual localization often must be based on the apparent position of features in the environment (landmarks). It is frequently the case that actual distance to landmarks is unknown and viewpoint must be determined from visual bearings alone. A number of relatively simple methods for solving this problem are available. However, visual bearings will never be completely accurate. As a result, it is important to have an understanding of the errors which can develop when using such methods. *A posteriori* analysis can predict the precision of an estimated viewpoint given expectations about the errors associated with the determination of visual bearings. Perhaps even more importantly, *a priori* analysis can be used to choose feature configurations which are least sensitive to error, thus yielding the most reliable localizations.

The best known method for doing localization is *triangulation* in which relatively simple trigonometric operations are used to determine viewpoint given the absolute bearings to two or more landmarks in known positions. However, absolute bearings are not always available and can be unreliable in many environments. For example, magnetic conditions in an area will affect compass readings. In such situations, alternative methods must be employed. It has been shown [Levitt *et al.*, 1987] that the visual angle between two landmarks constrains viewpoint to a closed torus–like surface. If a two–dimensional approximation of the environment is assumed, viewpoint is constrained to the boundary of a double circle (See Figure 1). When the landmarks can be ordered, viewpoint is restricted to one loop of that double circle. It follows that visual angles between three landmarks will constrain viewpoint to the intersection of three circles. Except in one special circumstance [Sutherland and Thompson, 1992], this uniquely determines the viewpoint. In this paper, we analyze the errors associated with this form of localization when the three landmarks are included within a visual angle of $< 180°$. Subsequent work will deal with errors which occur when the observation point is in the middle of a configuration of landmarks.

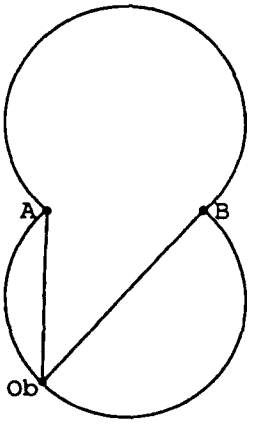


Figure 1: Angle from observer at Ob to landmarks A and B is the same from any location on the double circle.

We demonstrate a methodology for determining areas of uncertainty in localization given particular configurations of landmarks and errors in visual angle estimates to those landmarks. We then show the effects on localization of different configurations of landmarks when identical errors in estimate of visual angles are made. This *sensitivity* to error in visual angle estimate will vary considerably among configurations. The significant result is that feature configuration can dramatically affect the precision with which localization is accomplished. We are now able to specify criteria that aid in the selection of the best sets of features on which to base localization.

*This work was supported by National Science Foundation grant IRI-9196146, with partial funding from the Defense Advanced Research Projects Agency.

2 The Area of Uncertainty

Levitt *et al.* [Levitt *et al.*, 1987] point out that exact knowledge of visual angle between two landmarks constrains viewpoint to a partial circle which passes through the landmarks and that an error in visual angle estimate will constrain viewpoint to a thickened ring as shown in Figure 2. The thickness of the ring is determined by the amount of error in angle estimate. See [Sutherland and Thompson, 1992] for details of the computation.

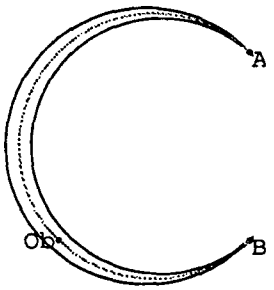


Figure 2: Error in visual angle constrains viewpoint to a thickened ring. A and B are the landmarks. The actual viewpoint, Ob, lies on the light circle.

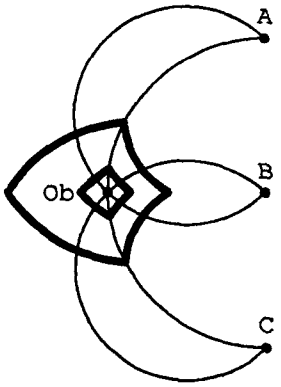


Figure 3: The dark lines surround the areas of uncertainty for errors of 10% and 30% in visual angle estimate.

When three landmarks are used, any given error in estimate constrains the viewpoint to the intersection of two rings.[1] In Figure 3, let α be the visual angle from the observer subtended by chord AB and β be the visual angle subtended by chord BC. The dark lines show the areas of uncertainty when the landmarks are in a straight line, the distance between landmarks equals the distance from the observer to the center landmark and both α and β measure 45°. The inner area represents an error less than or equal to ±4.5° or ±10% in both α and β. The outer area results when there is an error less than or equal to ±13.5° or ±30%.

Not only were angles α and β in Figure 3 identical in measure, but the error amount in each was the same. An observer will not necessarily make the same error in each angle measure. Let γ be the angle from the observer subtended by chord AC. Figure 4a shows the area of uncertainty for the 30% error shown in Figure 3. The area of uncertainty for angle γ is surrounded by dashed lines.

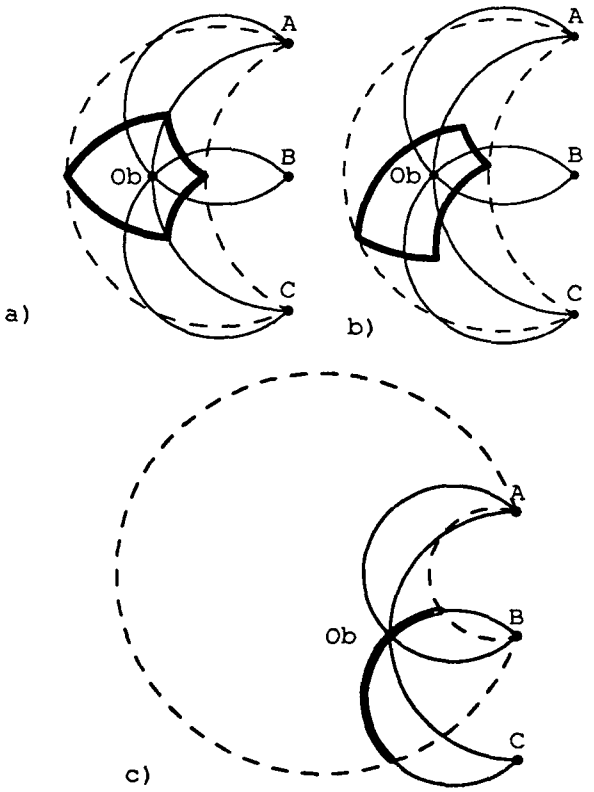


Figure 4: a) Dark lines surround the area of uncertainty for a 30% error with the same configuration as in Figure 3. Dashed lines surround the error area for angle γ. b) Dark lines surround the area of uncertainty for 40% error in α and 20% error in β. c) Error in estimate of angle α is 60%. Angle β is estimated perfectly. The result is a 30% error in estimate of γ. Dashed lines surround the error area for α.

Figure 4b shows the area of uncertainty for an error in α of 40% and error in β of 20%. Figure 4c shows the area of uncertainty for a 60% error in α and a perfect estimate of β. Note that the area is reduced to an arc of the circle passing through B and C. In general, if the error is additive with γ_e the error in γ, for any given γ_e, $\gamma_e = \alpha_e + \beta_e$ implies that $\alpha_e = \gamma_e - \beta_e$ for all β_e such that $0 \leq \beta_e \leq \gamma_e$. If the error is multiplicative, $\gamma_e = \alpha/\gamma * \alpha_e + \beta/\gamma * \beta_e$ implies that $\alpha_e = \gamma/\alpha * \gamma_e - \beta/\alpha * \beta_e$ for all β_e such that $0 \leq \beta_e \leq \gamma/\beta * \gamma_e$. In all cases, the resulting area of uncertainty equals the intersection of the two (possibly thickened) rings corresponding to the error in estimate of angles α and β. This intersection will always lie within the thickened ring corresponding to the error in γ, with the relationship of γ_e to α_e and

[1] A third ring passing through the two landmarks lying at greatest distance from each other can be computed, but it does not affect area size.

β_e as given in the above equations.[2]

3 Size of Area

Three factors affect the size of the area of uncertainty: amount of error in angle estimate, relative distance of observer to configuration and configuration shape.

Area size will increase with increased error. As previously described [Levitt *et al.*, 1987] each pair of landmarks produces a landmark–pair–boundary (LPB). These LPB's divide the plane into orientation regions. Crossing into a different orientation region changes the ordering of the landmarks. Since landmark order has been determined, it can be assumed that an LPB will not be crossed. Thus, the LPB puts a bound on the size of the area in the direction of overestimation of visual angle. However, there is no bound on how far back the observer can be located, causing total possible area of uncertainty due to amount of error in angle estimate to be unbounded.

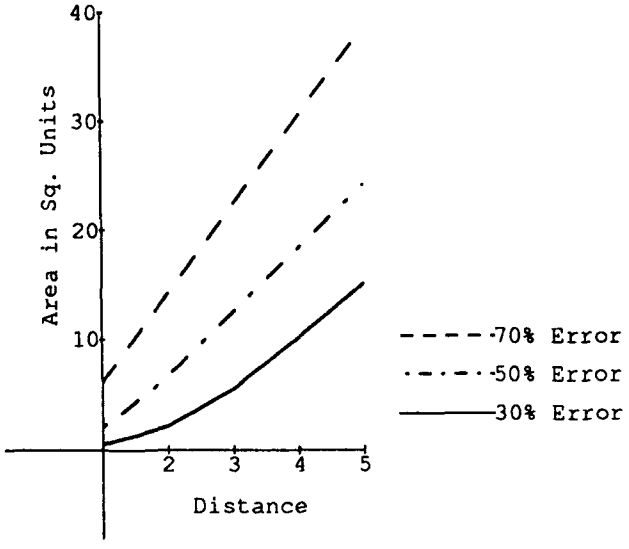


Figure 5: Size of area of uncertainty as observer moves away from the configuration.

Relative distance of observer to configuration will also affect area size.[3] Figure 5 shows the change in size of the area of uncertainty for a straight line configuration with landmarks one unit apart and observation point lying on the perpendicular bisector of the line joining the landmarks.

The third parameter to consider is the shape of the configuration. Thus far the configuration has been held constant with all three landmarks in a line and equally spaced. Comparative distance between straight line landmarks and angular relationship between non–linear landmarks will both affect area size.

In Figure 6, dark lines surround the area of uncertainty resulting with a 30% error in visual angle for the straight line configuration with landmarks A, B and C one unit apart and the observer five units from the configuration. The resulting areas with landmark C′ moved away from C are surrounded by dashed lines. C′ is 2 units from B on the left and 4 units away on the right. In both figures, the dark and dashed lines coincide where the boundary is determined by the error ring for the circle through A and B. A skewness develops on the boundary determined by the error ring for the circle through B and C. The area of uncertainty becomes smaller as C′ moves away from C.

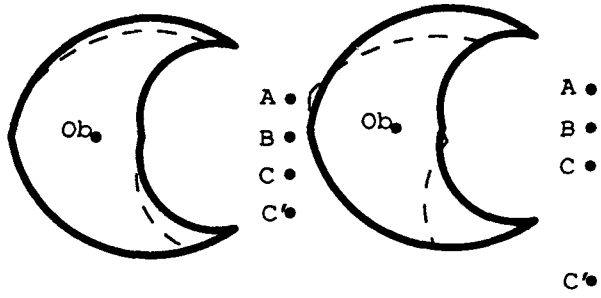


Figure 6: Both graphs show an error of 30% in visual angle with observer 5 units from configuration. The dark lines surround the area of uncertainty for the ABC configuration. The dashed lines surround the area for the ABC′ configuration.

The first step in analyzing how change in angular relationship between landmarks affects the size of the area of uncertainty is to consider the LPB's and resulting orientation regions. When all three landmarks lie on a straight line, the observer is constrained to a half plane. When the landmarks are not in a straight line, the three LPB's create seven orientation regions, as shown in Figure 7. The observer is constrained to one of those regions. The assumption will be made that the observer is not in the center region. Note that regardless of error size, the observer cannot move out of the orientation region. The area of uncertainty will always be within that region.

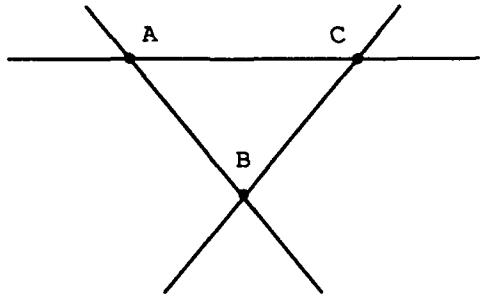


Figure 7: The LPB's in a non–linear configuration restrict the observer to one of 7 orientation regions.

[2]Although all graphs in this section show a straight line configuration of landmarks, the described conditions also hold for nonlinear configurations.

[3]Because the visual angles alone are used as a measure, distance is relative (e.g. distance of observer of 1000 feet to a straight line configuration with landmarks located 1000 feet apart is considered to be the same as a distance of 4000 feet with the landmarks 4000 feet apart).

If the locations of landmarks A and C are fixed and landmark B is moved along a line equidistant from A and C, the largest area of uncertainty for any give error amount will occur when B lies on the same circle as A, C and the observer. Although the navigator cannot tell by landmark map positions alone if the single circle condition exists, it can be ruled out if no part of the circle through the three landmarks lies in the observer's orientation region. This holds if, for example, the center landmark is closer to the observer than the other two. See [Sutherland and Thompson, 1992] for further analysis of the single circle configuration.

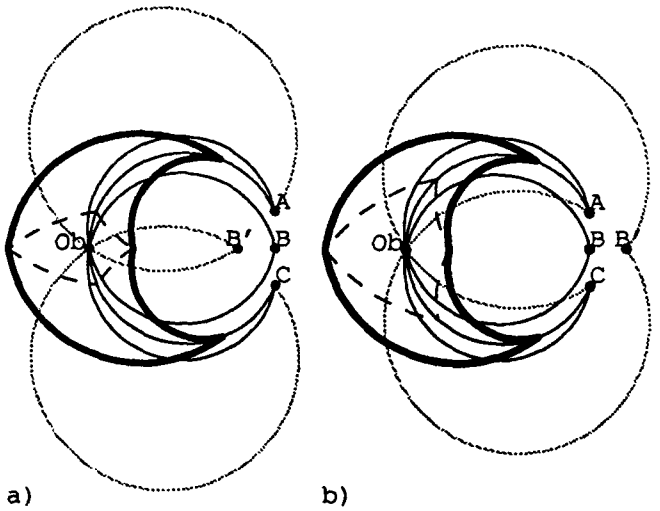


Figure 8: The heavy dark lines surround the area of uncertainty with 30% error in estimate of visual angle, landmarks A, B and C one unit apart and the observer 5 units away. The dashed lines in a) surround the area of uncertainty resulting when B′ is one unit closer to the observer than B. The dashed lines in b) surround the area of uncertainty resulting when B′ is one unit further away. The error circles are black for the linear configuration and grey for the non–linear configuration.

Figure 8 shows the area of uncertainty for a 30% error and the same basic configuration as in Figure 6 ($\alpha = \beta = 11.3°$). Landmark B′ is one unit closer to the observer in Figure 8a and one unit further away in Figure 8b. Although both show a decrease in area, the resulting area in Figure 8a is significantly smaller. Thus, the nonlinear configuration in Figure 8a is the least sensitive to error and would produce the most precise localization. Note that the length of the chord lying on the axis of symmetry does not change as landmark B is moved. For an observer facing the configuration, change in area is lateral only.

4 Shape of Area

Two shape properties must be considered. The first is symmetry. The second is eccentricity. If the configuration is symmetric, the observer is located on a line of symmetry of that configuration and the error in visual angle estimate is the same for both angle α and angle β, then the area of uncertainty will also be symmetric. If any of those properties do not hold, the area will not be symmetric.

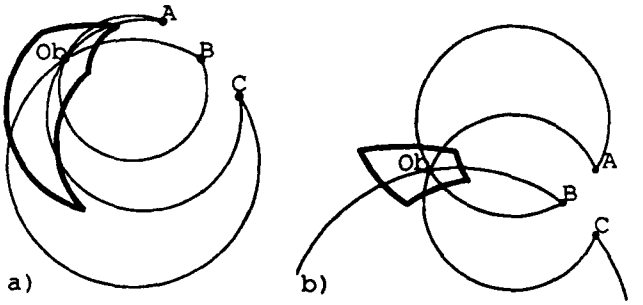


Figure 9: Shape of area of uncertainty is skewed when observer is moved off the line of symmetry of the configuration.

Figure 4 showed how unequal error in angle estimate affected shape. In Figure 6, landmark C was pulled away from the configuration. The area decrease resulted in an asymmetric shape. Figure 9 shows how the area shape is skewed when the observer is moved off the line of symmetry of the configuration.

The same three factors which affect the size of the area of uncertainty (amount of error in visual angle estimate, relative distance of observer to configuration and configuration shape) also affect the eccentricity. Future work will include analyzing how the conditions described in the next section relate to an area's eccentricity.

5 Conditions Affecting Sensitivity

We have identified two basic conditions which affect the sensitivity of a configuration. The first is the rate of change of visual angle measure as the observer moves in the environment. The second is the rate of change of the ratio of angles α and β as the observer moves.

Sections 3 and 4 contained several examples of the first condition. The navigator in Figure 8a needn't move far from the actual observation point to exit the area of uncertainty in the non-linear configuration because visual angles change at a much greater rate than they do with the linear configuration. The size of an area of uncertainty is based on this first condition. If the rate of change of visual angle with respect to viewpoint is high at a given location, a moving observer will leave the area quickly. If the rate of change is low, a significant distance could be traveled before the area boundary is reached.

In Figure 8, angles α and β have increased rates of change in the modified configurations. Figure 10 shows two three-dimensional graphs. Visual angle α is represented by the height of the surface. The rate of change of α is greatest where surface slope is steepest. An observer should make less error in localization when situated at a point of steep slope than at a point where the slope is shallow.

The second condition, the comparison of angle measure, also affects sensitivity. If, for example, an observer is located at a position such that angle α appears to be twice as large as angle β, it is unlikely that position will

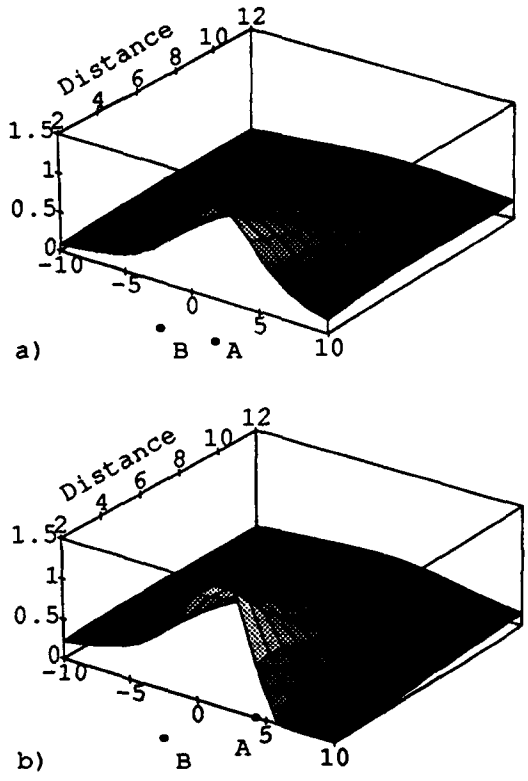


Figure 10: Visual angle α is represented by the height of the surface. Surface height is 0 at the LPB and outside of the orientation region. Distance is units to the furthest landmark. Landmarks are 4 units apart.

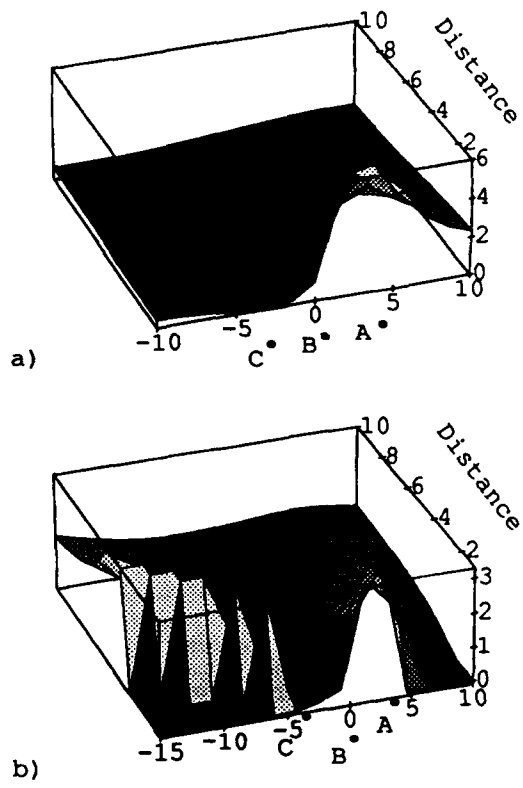


Figure 11: Visual angle ratio α/β is represented by the height of the surface. Height is 0 outside of the orientation region. Distance is units to the furthest landmark.

be estimated to be at a location where the visual angles appear to be equal. In the graphs of Figure 11, the ratio α/β is represented by the height of the surface. The maximum rate of change occurs at the points of steepest slope. An observer at those points should make less error in localization than if at a point of shallow slope.

To summarize, the rates of change of the visual angles and their ratio in an area around actual observer location depend on configuration shape and location of the observer with respect to the configuration. Large rates of change result in less sensitivity to error.

6 Conclusions

At this point in our analysis:

- We can determine size and shape of the area of uncertainty for any configuration of three landmarks, observer position and error in visual angle.
- We are able to predict how that area will change in size and shape as configuration shape changes.
- We have isolated properties which can be used to predict magnitude and type of errors in localization.
- We have identified two basic conditions which are affected by changes in the configuration and, in turn, affect size and shape of the area of uncertainty.

Future work will include:

- Determining what properties can be used for localization if the observer is in the center of the configuration.
- Analyzing how the rate of change of angle ratio relates to the eccentricity of the area of uncertainty.
- Defining a measure of sensitivity for any given configuration, including a weighting of the properties which affect area size and shape.

References

[Levitt *et al.*, 1987] T. S. Levitt, D. T. Lawton, D. M. Chelberg, and P. C. Nelson. Qualitative navigation. In *Proc. DARPA Image Understanding Workshop*, pages 447–465, Los Altos, CA, February 1987. Morgan Kaufmann.

[Sutherland and Thompson, 1992] K. T. Sutherland and W. B. Thompson. Sensitivity of feature configuration in viewpoint determination. Submitted to the IEEE Conference on Computer Vision and Pattern Recognition, June, 1992.

Strategies for Localization*

Marian R. Heinrichs,
Kip Smith, and
Herbert L. Pick, Jr.
Center for Research in Learning,
Perception, and Cognition
University of Minnesota
Minneapolis, MN 55455

Bonnie H. Bennett
Computer Science Dept.
University of Minnesota
Minneapolis, MN 55455

William B. Thompson
Dept. of Computer Science
University of Utah
Salt Lake City, UT 84112

Abstract

Landmark-based navigation is a problem solving activity in which correspondences must be established between map and view features, and then current location determined from these correspondences. A variety of high-level strategies are important in order to control combinatorial complexity and minimize difficulties due to problems in accurately recognizing different features. These strategies organize features into distinctive configurations, control the order in which correspondences are established, and regulate the manner in which viewpoint hypotheses are generated and validated.

1 Introduction

Localization is an essential aspect of almost any map-based navigation activity. Localization involves the determination of the current viewpoint given one or more views, information from a map, and if available, a history of past movement and observations. While the image understanding community has studied the problem of matching maps and aerial imagery in some depth (e.g., [McKeown and Denlinger, 1984]), less attention has been paid to localization problems involving outdoor, ground-level viewpoints. Much of the ground-level work that has been done has been directed at lower-level problems such as road following (e.g., [Thorpe *et al.*, 1988]).

Landmark-based localization establishes correspondences between visually distinct landmarks and topographically or culturally distinct map features and then infers a viewpoint based on the geometric constraints imposed by these correspondences. Work done on indoor mobile robotics has tended to assume that landmarks could be uniquely identified (e.g., [Chatila and Laumond, 1985]). Outdoors, topographic features are not so easily distinguished. Difficulties arise due to ambiguity in establishing correspondences and combinatorial complexity due to the large number of potentially relevant features that are often present.

We have previously argued that localization and feature-based object recognition have a common computational form [Thompson *et al.*, 1990]. In object recognition, various forms of heuristic search have proven to be quite powerful in controlling combinatorics [Grimson, 1990, Grimson, 1991]. These same optimizations are relevant to localization. In addition, unique aspects of the outdoor localization problem allow the use of powerful high-level problem solving strategies to guide the establishment of correspondences and the estimation and verification of viewpoints.

2 Strategies

Computational analysis, computer simulations, and experiments done with expert map users all point towards a small set of strategies being critical to the solution of difficult localization problems. These strategies are relevant to localization approaches which operate by establishing correspondences between features in the view and on the map, using these feature correspondences to hypothesize one or more possible viewpoints, and then using some sort of verification process to evaluate viewpoint hypotheses [Thompson *et al.*, 1990, Smith *et al.*, 1991].

Concentrate on the view first.

Localization problem solving should be initiated by a period of general visual reconnaissance, focused on the terrain view in preference to the map. In general, available maps cover an area much larger than can be seen. As a result, the majority of map features will not be relevant to any particular viewpoint determination, whereas most of the distinctive visual features will have correspondences on the map. One of the major sources of difficulty in object recognition is "clutter" consisting of image features not associated with the object of interest. Matching is therefore usually driven by object models which contain only features relevant to a particular object (e.g., [Grimson, 1990]). In localization, the clutter is in the "model" (i.e., map). As a result, the search for feature correspondences should start from the view.

Landmark features should be organized into configurations.

Most outdoor environments have a large number of terrain features that are potentially relevant as landmarks. As a result, the combinatorial complexity associated with establishing possible correspondences between map and view features is large. This complexity can

*This work was supported by National Science Foundation grant IRI-9196146, with partial funding from the Defense Advanced Research Projects Agency.

be reduced by first assembling small groups of nearby features into *configurations* before correspondence is attempted. Not only is the number of landmarks reduced, configurations are more likely to be distinctive than individual features, reducing the ambiguity in matching. A particularly useful approach is to assemble configurations formed along the line of sight. Such configurations have the viewpoint independent property that they are necessarily organized in a linear fashion on the map as well as in the view. In addition, they have the viewpoint dependent property that once found on the map, they constrain the viewpoint to a line. When possible, several such configurations surrounding the viewpoint should be found. If this is not possible, then features which are at least connected to one another should be used in developing configurations, since these are far easier to find on a map.

Information about terrain at the viewpoint is important.

This may seem obvious, but in fact most approaches to landmark-based robot navigation do not pay special attention to local features of the immediate environment. If it is possible to determine that an agent is at or near a particular terrain feature type, then the viewpoint is constrained to be at one of the corresponding types on the map. Determination of local feature type is often easier than evaluation of more distant features. For example, active range sensors can sense whether the viewpoint is on a ridge but are inoperative over the larger distances to major landmarks such as distant peaks.

Often, localization is possible by assembling a configuration of features that includes the immediate area and several nearby landmarks with distinctive relationships to each other and to the viewpoint. In such cases, a simple search for comparable configurations in the map can generate a viewpoint hypothesis without any need for triangulation or more sophisticated geometric reasoning.

Multiple hypotheses need to be generated and examined.

Heuristic search aims at focusing analysis to quickly select a viewpoint hypothesis that can be evaluated against the current view of the environment. Terrain features are highly ambiguous, however, and so it is difficult to identify landmarks with certainty. Any single viewpoint hypothesis based on a small number of features has a high probability of being incorrect. In complex terrain, it appears to be necessary to develop a number of different plausible hypotheses for the viewpoint before verification takes place.

Hypotheses should be compared using a disconfirmation strategy.

Validation of a hypothesis involves a comparison of the actual view with expectations generated from the map based on the presumed viewpoint. In performing this comparison, it is most important to note expectations that are *not* met. If one clear mismatch is found, then the associated hypothesis should be eliminated. Since terrain tends to have lots of features that more or less look the same, validation based on finding expected features is far less effective than rejecting hypotheses when expected features are not found.

The ability to move to alternate viewpoints is important.

Movement to bring obscured features into view or to generate parallax sufficient to gain distance estimates has clear advantages when solving localization problems. Often overlooked, however, is the importance of movement in verifying hypothesis. As previously mentioned, recognizing terrain features when you are standing on them is usually easier than recognizing them from a distance. When possible, viewpoint hypothesis should be used to generate expectations about nearby features which can then be confirmed or disconfirmed by moving to the predicted location of the feature.

3 Expert Map Users

Experiments with expert map users show the importance of the above strategies. Experienced topographic map readers were given difficult field problems and asked to describe their procedures as they attempted to work out the solutions. Their verbal protocols were recorded and analyzed to provide a detailed description of the localization process. Since the ability to judge slope and distance is commonly thought to be central to using topographic maps, followup investigations of people's ability to judge these properties were also conducted.

The map readers who participated in this study had extensive professional or recreational experience using topographic maps. Included were geologists, wilderness guides, nationally ranked orienteers, and members of the military. The study was done in two areas of east-central Minnesota, in terrain characterized by rolling hills. There were no distinctive landmarks that would allow for a quick solution through simple recognition. The problem was a difficult one, and only 12 of 41 subjects arrived at a correct solution. All of those who successfully solved the problem employed the first four strategies described in section 2, and many also used movement to assist in verification. One or more of these four strategies was absent in the problem solving approaches used by those who were not able to correctly determine the viewpoint.

Figure 1 provides a high-level schematic of a portion of the protocol taken from one of the successful experts. A detailed analysis of this and other protocols can be found in [Pick *et al.*, 1992]. Fragments of text from the actual transcript illustrate many of the strategies described in the previous section:

Concentrating on the view first and then searching for correspondences on the map.

> "[early in the protocol, starting with the immediate area] All right, well I noticed I'm at one of the higher points within this area, so that's important. [Now to the map] So I'm first looking on the map, for some higher points on the map. [Next refine the description of the immediate area] Now I'm just taking a look around. It seems like I'm at the top of a hill or nearly at the peak of the hill."

Organizing features into configurations.

> "I'm at a high point. Directly north is a fairly flat area and north of that it gets steep and then there's a lake..."

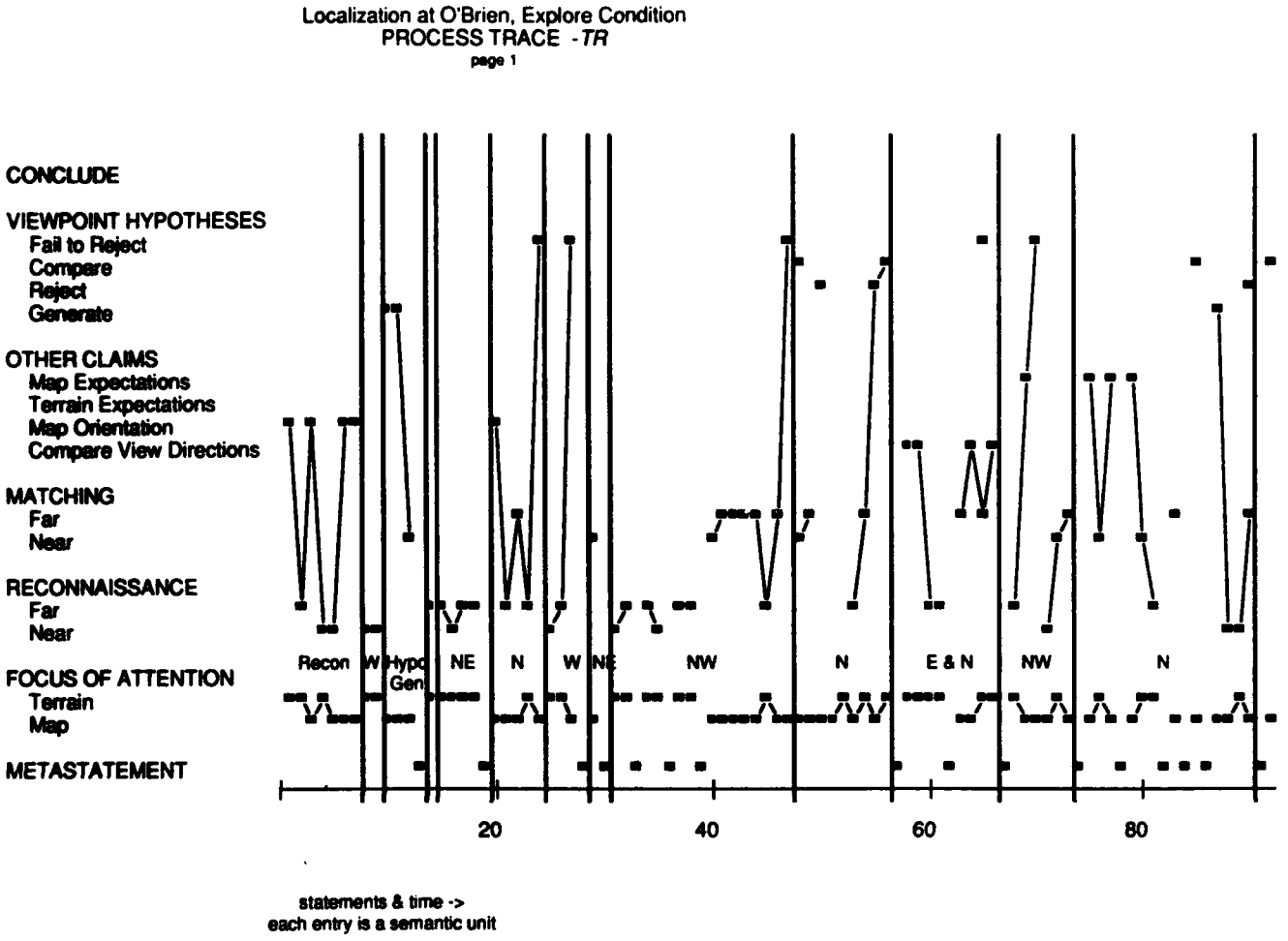


Figure 1: Partial process protocol trace.

Multiple hypotheses need to be generated.

> "Um, for example say, somewhere here [points on the map to what is in fact the correct hypothesis] or on a hill here [points to an incorrect viewpoint hypothesis]."

Comparing hypotheses using a disconfirmation strategy.

> "OK, I still kind of like this area [points to correct hypothesis]. But then I was looking up on the map. I also have a high here [points to an incorrect hypothesis] with a pond that's not very far at all. It just doesn't seem to work well because there's a fairly steep and long gradient here before you get to a flat part. And I don't see that where we're standing."

The last quotation illustrates the tendency of experts to use qualitative rather than quantitative judgments of magnitude both when referring to the map and when considering the terrain. This tendency is so striking it led us to conduct an investigation to determine whether or not quantitative judgements about slope and distance could accurately be made by people in terrain similar in topography and extent to that of the localization problem. Observers were very poor at estimating both distance and slope. In particular, distance judgments along a line of sight were underestimated in comparison with judgments perpendicular to the line of sight, while slopes were dramatically overestimated. The errors are reduced but not eliminated by restricting judgments to more level and homogeneous terrain. These results are of engineering relevance, since image understanding systems also are poor at determining distance and slope unless active sensing is used. It is likely that the strategies observed with expert map users are adapted to solving localization problems using primarily qualitative information about the layout of the visible terrain.

4 Implementation

We are constructing a computer program capable of reasoning about localization problems in a manner similar to that described above. This program allows us to validate the utility of different problem solving strategies

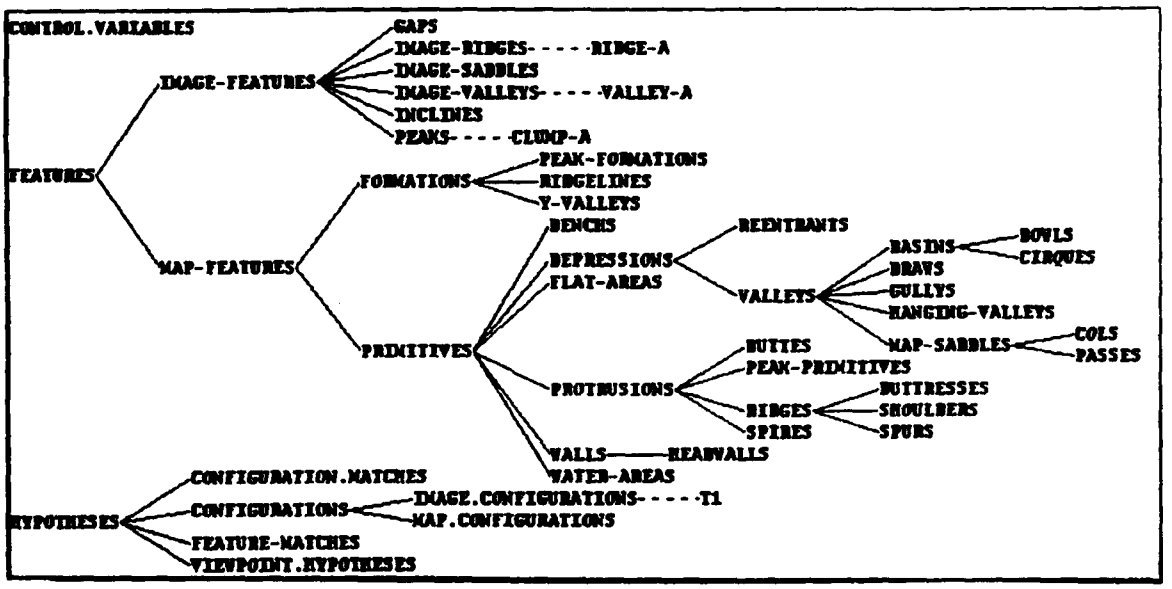


Figure 2: Taxonomy of image and map features.

by experimenting to determine the efficiency and accuracy of particular approaches. In its current form, the program uses many of the strategies described above together with a number of techniques which reduce complexity and ambiguity, while quantifying the uncertainty that remains.

	Image-Features					
	Gaps	Image-Ridges	Image-Saddles	Image-Valleys	Inclines	Peaks
Map-Features						
Benches	0	3	0	0	1	1
Depressions	5	0	5	5	0	0
Reentrants	5	0	1	3	0	0
Valley	5	0	3	5	0	0
Basin	5	0	5	5	0	0
Bowls	3	0	3	5	0	0
Cirques	3	0	3	5	0	0
Draws	5	0	1	5	0	0
Gullies	5	0	1	5	0	0
Hanging-valleys	3	0	3	5	0	0
Map-Saddles	3	0	5	3	0	0
Cols	3	0	5	5	0	0
Passes	3	0	5	5	0	0
Protrusions	0	3	0	0	3	3
Buttes	0	3	0	0	3	3
Peak-primitvs.	0	3	0	0	3	5
Ridges	0	5	0	0	3	1
Buttresses	0	5	0	0	3	3
Shoulders	0	5	0	0	3	3
Spurs	0	5	0	0	3	3
Spires	0	3	0	0	3	5
Walls	0	3	0	0	5	0
Headwalls	0	3	0	0	3	0

Figure 3: Compatibility of map and image features.

Hierarchical matching is used to establish correspondences between map and image features. The hierarchy is based on a taxonomy of geographic terms and represents the classes, subclasses, and instances of image and map features (see Figure 2). Image features, for example, are divided into the classes of gaps, ridges, saddles, valleys, inclines and peaks. Map features are organized into a richer structure, since more information is available about them from lower-level, map-understanding processes. This taxonomy provides the conceptual catalog on which the representations for the problem-specific data are built. The class, subclass, and instance relations form the basis for the proximity of concepts in the hierarchy. Proximity is used as a criteria for the value of matches between compatible, but not identical features. Figure 3 shows the *a priori* likelihood currently assigned to matches between image and map features, with 5 indicating that the features are highly compatible and 0 indicating that the features are very unlikely to match.

Three types of hypotheses are used: feature match hypotheses, configuration match hypotheses, and viewpoint hypotheses. These represent the possibility that a feature or a configuration from the image corresponds to a feature or a configuration from the map, or that the agent is at a given location on the map, viewing the world from a certain direction. Hypotheses are also implemented in a hierarchy, reflecting the concrete and isolated nature of the low-level data and the abstract and conglomerate nature of the high-level data. Viewpoint hypotheses are made up from and depend on configuration match hypotheses. These in turn are made up from and depend on configurations and feature match hypotheses which themselves rely on the incoming image and map data.

The hierarchical organization of features and hypotheses facilitates advanced control of the reasoning processes. Procedures which combine low-level data into higher-level components are data-driven. Other procedures which use indications from high-level data to drive their analysis of the low-level data are hypothesis-driven. An advanced, problem-solving control structure integrates the data- and hypothesis-driven reasoning com-

ponents to balance their work.

The following example illustrates several of the features currently implemented. The problem corresponds approximately to Figures 1 and 2 in [Savitt *et al.*, 1992]. The output contains fragments from an actual run of the program, simplified to improve readability. A much more detailed description of this same example is presented in [Bennett, 1992].

Starting with the view, initial reconnaissance generates prominent features:

```
(((feature peak-a) (type image) (isa image-peak))
  ((feature valley-a) (type image) (isa image-valley)
    (left-of peak-a))
  ...)
```

These features are evaluated to locate configurations:

```
((image-configuration t1)
  (components (ridge-a valley-a peak-a))
    (relation left-of))
```

Additional reconnaissance generates prominent map features:

```
(((feature Mount-Moran) (type map) ... )
  ((feature Mount-St-John) (type map) ... ) ...)
```

Map configurations are identified:

```
(((map-configuration t10)
  (components (m-wister g-teton m-st-john))
    (relation north-neighbor)) ... )
```

Eight feature matches are hypothesized:

```
(((feature-match t2) (image-feature peak-a)
  (map-feature m-moran) ...) ...)
```

Effective configurations are not found, forcing a search for configurations among features at finer levels of detail:

```
((config-match t128) (matches ((ridge-a the-jaw)
  (valley-a paintbrush-c) (peak-a m-moran))) ... )
```

Twelve viewpoint hypotheses are proposed:

```
((viewpoint-hypothesis t129)
  (matches ((ridge-a the-jaw)
    (valley-a paintbrush-c) (peak-a m-moran)))
  (viewing-from east) (viewing-location (...))
...)
```

Hypothesis-driven reasoning explores each of the proposed viewpoints, refining the feature matching and location hypotheses. Eventually, the correct answer surfaces as the best hypothesis. Other hypotheses, less consistent with the data are ranked accordingly.

The program is currently operating with simulated input corresponding to a portion of Grant Teton National Park (see the example in [Thompson *et al.*, 1990]). We are in the process of using methods described in [Savitt *et al.*, 1992] to allow operation on real data.

5 Conclusions

Over the course of our research, we have obtained a number of important insights into effective and efficient methods for solving localization problems. These results have been made possible by an interdisciplinary approach which integrates computational analysis, investigations of expert map users, and computer simulations. Problem solving strategies have been discovered which appear to give substantial leverage in managing the complexity and ambiguity associated with navigation in outdoor terrain. The work has applications in both automated navigation aids and in training. In addition, there is likely to be relevance to other combinatorial geometric matching problems such as feature-based object recognition.

References

[Bennett, 1992] B. H. Bennett. *A Problem Solving Approach to Localization.* PhD thesis, University of Minnesota, 1992.

[Chatila and Laumond, 1985] R. Chatila and J.-P. Laumond. Position referencing and consistent world modeling for mobile robots. *1985 IEEE International Conference on Robotics and Automation*, pages 138–145, March 1985.

[Grimson, 1990] W. E. L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints.* The MIT Press, 1990.

[Grimson, 1991] W. E. L. Grimson. The combinatorics of heuristic search termination for object recognition in clutered environments. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13:920–935, September 1991.

[McKeown and Denlinger, 1984] D. M. McKeown, Jr. and J. L. Denlinger. Map-guided feature extraction from aerial imagery. *IEEE Workshop on Computer Vision: Representation and Control*, pages 205–213, 1984.

[Pick *et al.*, 1992] H. L. Pick, Jr., M. R. Heinrichs, D. R. Montello, K. Smith, C. N. Sullivan, and W. B. Thompson. Reading topographic maps In J. Flach, P. Hancock, J. Caird, and K. Vicente, editors, *Ecology of Human-Machine Systems.* 1992.

[Savitt *et al.*, 1992] S. L. Savitt, T. C. Henderson, and T. L. Colvin. Feature extraction for localization. In this proceedings, 1992.

[Smith *et al.*, 1991] K. Smith, M. R. Heinrichs, and H. L. Pick, Jr. Similarity judgment and expert localization. In *Proceedings Thirteenth Annual Conference of the Cognitive Science Society*, August 1991.

[Thompson *et al.*, 1990] W. B. Thompson, H. L. Pick, Jr., B. H. Bennett, M. R. Heinrichs, S. L. Savitt, and K. Smith. Map-based localization: The "drop-off" problem. In *Proc. DARPA Image Understanding Workshop*, pages 706–719, September 1990.

[Thorpe *et al.*, 1988] C. Thorpe, M. H. Herbert, T. Kanade, and S. A. Shafer. Vision and navigation for the Carnegie-Mellon Navlab. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10:362–373, 1988.

Feature Extraction for Localization*

Steven L. Savitt
Computer Science Department
University of Minnesota
Minneapolis, MN 55455

Thomas C. Henderson and
Thomas L. Colvin
Department of Computer Science
University of Utah
Salt Lake City, UT 84112

Abstract

Visual navigation with a map requires effective perception of both the information in the map as well as the terrain imagery that is visible from the current position. In open, outdoor terrain where cultural features are lacking, ridgeline and valley features are particularly important. This paper describes a novel method for extracting such features from digital elevation maps. In addition, preliminary results are presented on extracting corresponding features from views of the terrain.

1 Introduction

Determining the location of a viewpoint with respect to a map requires two perception tasks: a map understanding process which extracts corresponding features from the map and an image understanding process which extracts salient landmarks from one or more views. Peaks, ridgelines, saddles, and valleys are important landmarks in outdoor terrain when cultural features in known locations are absent [Pick *et al.*, 1992]. These features have precise definitions, specified in terms of local slope properties of the underlying surface [Peucker and Douglas, 1975, Haralick *et al.*, 1983]. Using these definitions to extract navigational features from elevation data can be difficult, however, due to their intrinsically local nature. The saliency of map features depends on properties over a variety of scales [Savitt, 1991]. In subsequent sections, we outline an alternate approach for finding ridgelines and valleys given digital elevation data. We then conclude with some preliminary results on the extraction of the same features from ground level imagery of outdoor terrain.

2 Extraction of Map Features

Valley features can be efficiently found using a "reverse engineering" strategy which simulates aspects of the valley formation process. Valleys are erosional features. We model the this erosion using a hydrologic simulation, though the method will work for lands shaped by other processes. Since the eroding flows moved from higher to lower elevations, the floor of the resulting valley varies monotonically in elevation along its length. Also the width, depth, and hence visual significance of a valley tends to relate to the volume of the hydrologic flow that formed the valley. Fluid flows over the region of interest are simulated to reveal the paths through which the eroding flows were channeled. Then the location and magnitude of the simulated flows are used to determine the location and saliency of the resulting valley features. The approach was motivated by the early work of [Speight, 1968] who developed a manual technique for extracting watercourse features from printed contour maps. A computer implementation of Speight's algorithm for hydrologic flow analysis was compared by [Mark, 1983] to the algorithm of [Peucker and Douglas, 1975] which is based on an analysis of local surface properties.

The implemented algorithm consists of a two step process in which a hydrologic flow simulation is first used to determine how much fluid would pass each point in the elevation grid if the terrain surface were subjected to a uniform rainfall. Then in a second step the desired features are extracted from the resulting flow analysis, and the relative flow strengths associated with each feature are used to predict the visual saliency of the feature.

The hydrologic flow simulation operates on a data structure called a flow image which consists of flow counter cells that have a one to one correspondence with the elevation samples in the terrain map. These cells are used to keep track of the total amount of fluid that flows past the corresponding surface point of the terrain.

The simulation is initiated by introducing one unit of fluid into each cell of the flow image. Then beginning with the surface element having the highest elevation, the fluid in the associated element is passed down to the nearest of the eight neighboring elements with the steepest downward slope from the initial element.[1] The passing is accomplished by adding the amount of fluid being passed to the quantity of fluid already registered in the flow counter of the receiving element. The element from the cell with the next lower elevation value is pro-

*This work was supported by National Science Foundation grant IRI-9196146, with partial funding from the Defense Advanced Research Projects Agency, and by a grant from Texas Instruments.

[1] Since elevation values are quantized and of limited range, an efficient bin sort can be used.

cessed next, and the process continues until the lowest elevation point is reached. The resulting flow image reveals how much of the initial fluid flows past each point of the terrain. The values can range from a minimum of one unit which is always the case for the highest point in the terrain, to a maximum of $n \times m$ units in the rare case that all of the initial fluid distribution is channeled downward past a single point of the terrain. The magnitude of flow values in the flow image are used to predict the visual salience of the associated valley features. Our experience indicates that the flow magnitude is a good predictor of visual salience as it reveals the intensity of the valley formation process which caused certain valleys to become more visually salient than others in terms of their extent, depth and breadth.

Ridge features are found in a similar manner. In effect, the surface of the earth is considered to be an infinitely thin shell and when inverted, the valleys become ridges and the ridges become the valleys. The hydrologic flow based feature extractor is then applied to the inverted terrain model to find ridgelines. Ridgelines, of course, were not formed in this manner and in fact have certain key properties that are quite distinct from valleys. In particular, ridgelines are seldom monotonic in elevation along their length. Nevertheless, simple modifications to the hydrologic flow algorithm provide a useful mechanism for localizing ridges.

The apparent success of the flow simulation technique for extracting ridgelines can be explained as follows. Ridges that extend radially outward from the peak of a hill tend to have extended sections which do decrease monotonically in elevation. Once inverted, these ridgeline sections are then directly detectable by the valley detection algorithm since they present a continuous channel through which a simulated fluid flow can pass. The terminations of the monotonic sections of ridgelines occur at either local peaks or saddle points. The local peaks will cause the simulated flow from the two branches on opposite sides of the peak to join at the site of the peak resulting in the ridgeline being properly identified as a continuous feature through the peak. The problem occurs at the locations of the saddle points that exist along the ridgeline between pairs of local peaks. When encountered during the simulation, a peak blocks the simulated flow, and that the flow must be purposefully extended across an adjacent saddle point along the ridge to some point at a higher elevation than that of the peak.

The magnitudes of simulated flows along a ridgeline are used as an indication of the visual salience of the ridgeline, although the rationale for doing so differs from the argument forwarded for the case of valley features. In the case of ridges, the magnitude of the simulated flow is not directly related to the strength of the process that formed the ridges, as is the case for valleys. However the magnitude of the fluid that impinges on a ridgeline is directly related to the surface area of the terrain that extends from the ridgeline to the interposing valleys on either side of the ridge, and this surface area is an important metric for estimating the visual saliency of a ridge. The surface area can be thought of as the product of the length of the ridge and the slant distance between the ridgeline and each of the adjacent valleys, both factors which contribute to the visual salience of a ridgeline.

After the flow image has been generated, an adaptive threshold is applied to extract the desired terrain features. In the case of valleys and ridges, the features emerge in a dendrite-like tree pattern that depicts the connected valley and ridge networks. In the final phase of the algorithm, the resulting network trees are parsed to extract the individual branches of the network, and the strength of the fluid flow associated with each branch is used to determine the visual saliency of the terrain feature. The individual valleys and ridges are then organized into a hierarchical database of terrain features. The terrain features with the highest salience are placed in the highest position within the hierarchy.

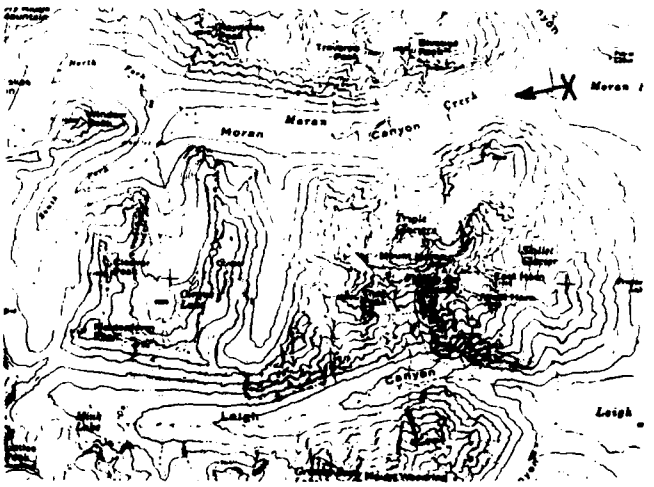


Figure 1: Topographic map.

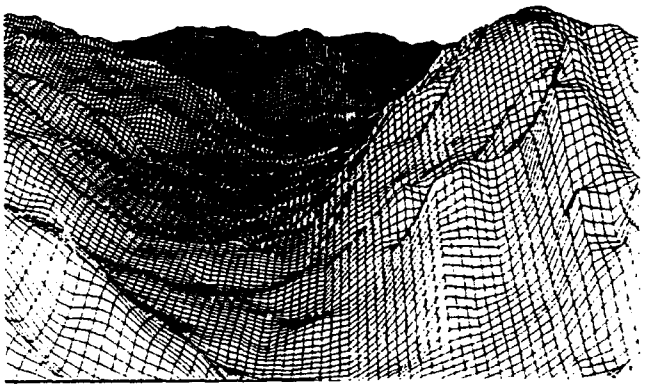


Figure 2: Wireframe view.

Figure 1 shows a conventional topographic map of a region in Grand Teton National Park. Figure 2 shows a simulated view, derived from digital terrain data, of the region as seen from the position marked with the "x" in Figure 1. Figure 3 illustrates the terrain features extracted using local slope behavior of the terrain surface. The white lines are the valley features and the black lines



Figure 3: Ridgelines and valleys using local slope properties.

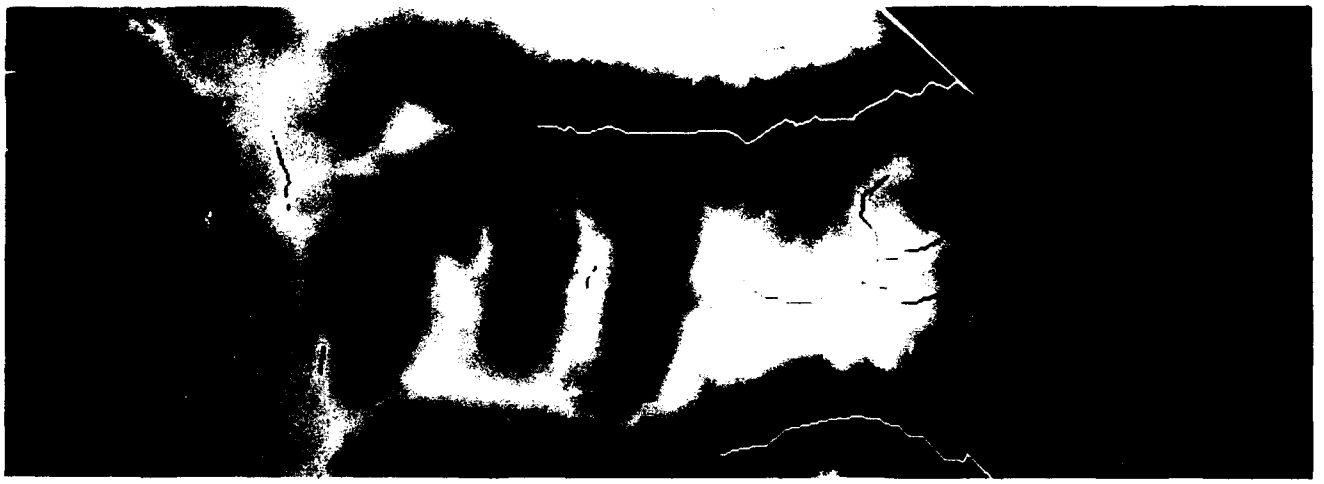


Figure 4: Ridgelines and valleys using hydrologic simulation with high threshold.

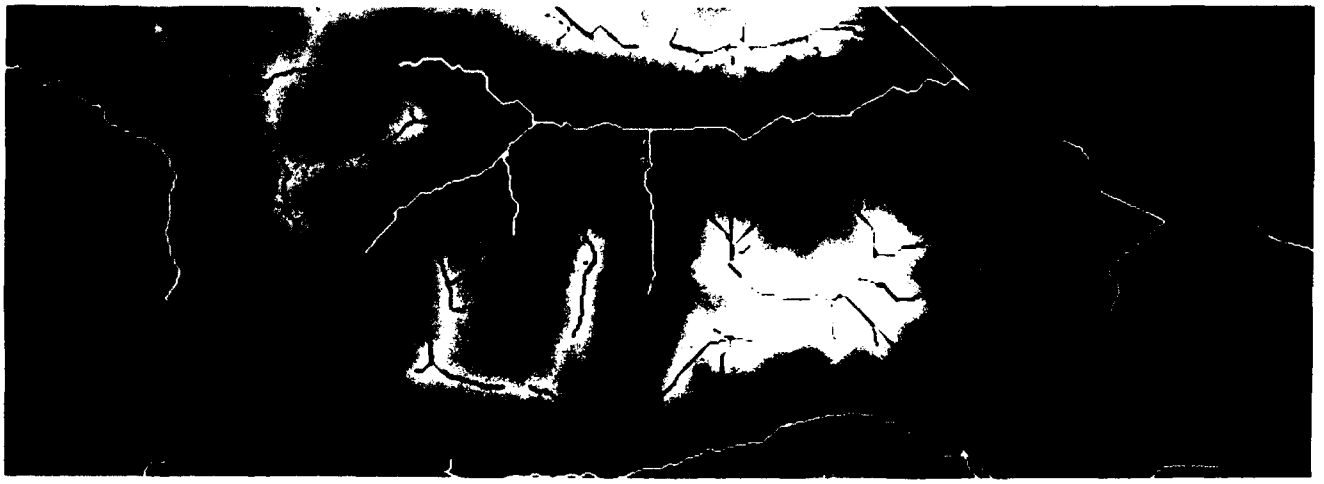


Figure 5: Ridgelines and valleys using hydrologic simulation with low threshold.

are the ridge features. The features are overlaid on an intensity coded elevation image in which darker shades of grey correspond to pixels at lower elevations.

The linear terrain features that are extracted by the local slope method consist of several partial feature segments along the extent of the full feature. Consider, for example, the valley feature along Moran Canyon. The valley is readily apparent in the wireframe representation due to the presence of the two enclosing hillsides. The valley floor tends to be rather flat, irregular and winding, with the result that a process to extract the linear valley feature based on local slope information results in the detection of only partial spans of the full valley. As seen in Figure 3, the extracted valley feature is broken into approximately ten segments in the span before the fork, and approximately 40% of the valley is missed. A second problem with local slope based feature detection is that visual saliency and navigational significance of terrain features does not appear to correlate well with measures of local slope. For instance, one of the more dominant valley features that is detected by the local slope based feature detector is a side valley that branches south off of Moran canyon between Thor Peak and Cleaver Peak. The narrower, more distinct valley bottom of this side valley results in a stronger detection result than that obtained for the primary valley of Moran Canyon even though Moran canyon is clearly the more visually salient terrain feature in the region.

The result of the flow based algorithm is shown in Figures 4 and 5. As before, the white lines are the valley features and the black lines are the ridge features. In Figure 4 a rather high threshold is used, leaving only the more prominent features. Figure 5 uses a lower threshold and thus includes greater detail. Notice that the valley features are now fully connected as a result of the flow continuity feature of the underlying simulation. In addition, the intensity of the flow provides a metric that can be used to predict the visual and navigational saliency of the respective feature. For example, Figure 4 includes only the main valleys while in Figure 5 the less salient side valleys emerge.

3 Extraction of Image Features

Ridgelines in terrain appear as distinct contours with a predominately horizontal orientation in ground-level views of the terrain. Valley features are harder to extract from terrain images, since much of the valley itself is typically occluded by the ridges to either side. When enough of the walls are visible, texture measures can sometimes be used to locate valleys [Thompson *et al.*, 1990]. Otherwise, the existence of a valley must be inferred from the ridgeline pattern. While the possible associations between ridge shape and valleys are both complex and ambiguous, it is generally the case that smooth dips and T-junctions in ridgelines provide evidence for the existence of valleys [Bennett, 1992].

We are currently investigating how conventional image segmentation techniques can be adapted to the unique requirements associated with the extraction of ridgeline features. The problem is difficult because the irregular nature of terrain precludes the effective use of simple line-finder type algorithms and because of the presence of extraneous features such as sky, clouds, trees or other vegetative cover. Results presented here only involve the sky-terrain boundary. This is the easiest set of ridgelines to find and can aid in locating other ridgelines closer to the viewpoint.

Figure 6 is a digitized image taken from a location near the Alta ski area, just outside of Salt Lake City. Figure 7 shows the output of a standard zero-crossing edge detector applied to the image shown in Figure 6. Note that while sky-terrain boundary elements have been found, many long edge segments exist above and below.

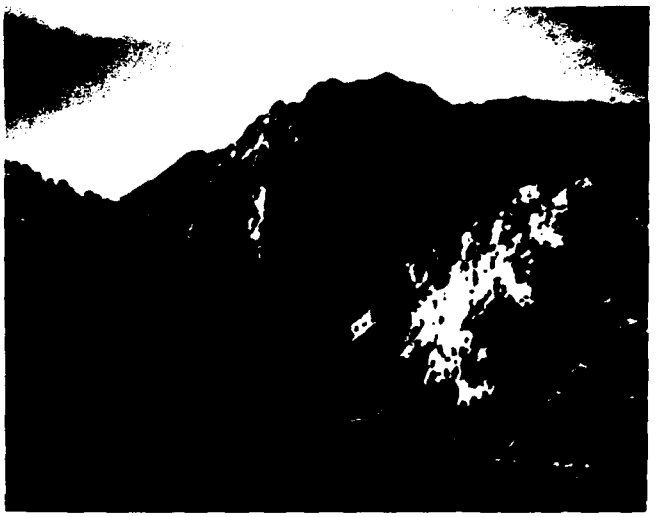


Figure 6: Outdoor terrain near Salt Lake City.

Figure 8 shows the output of an edge filtering algorithm which produces a connected contour across the image. The contour is determined using a heuristic search algorithm that attempts to simultaneously minimize the contour length and the number of gap pixels that have to be filled. (Similar methods date back at least as far as [Martelli, 1972]). In this example and other similar tests, the algorithm was effective in finding the actual sky-terrain boundary. For comparison purposes, Figure reffig:ground-truth shows the predicted occlusion boundaries at approximately the viewpoint from which Figure 6 was taken, generated from digital elevation data of the area. Figure 10 shows the results of applying a peak and saddle detector to Figure 8. Currently, this is done using a simple extrema detector. Work is underway to characterize the shape of extracted features and to differentiate between true saddle points and T-junction gaps indicating that one peak is in front of another.

References

[Bennett, 1992] B. H. Bennett. *A Problem Solving Approach to Localization*. PhD thesis, University of Minnesota, 1992.

[Haralick *et al.*, 1983] R.M Haralick, L.T. Watson, and T.J. Laffey. The topographic primal sketch. *Int. Journal of Robotics Research*, 2(1), Spring 1983.

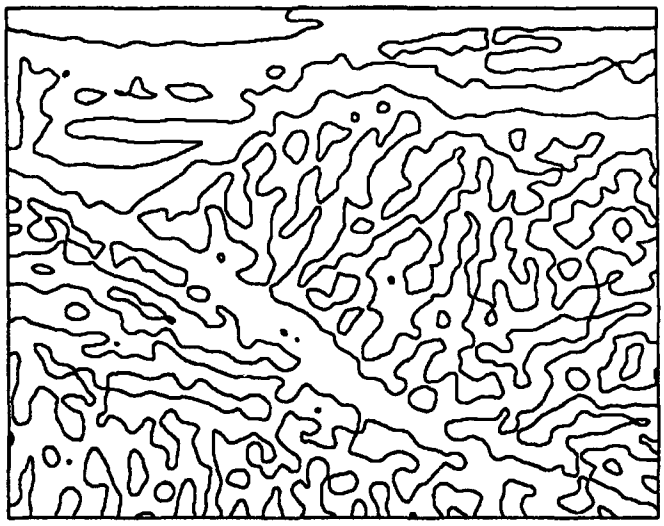


Figure 7: Edge detector applied to Figure 6.

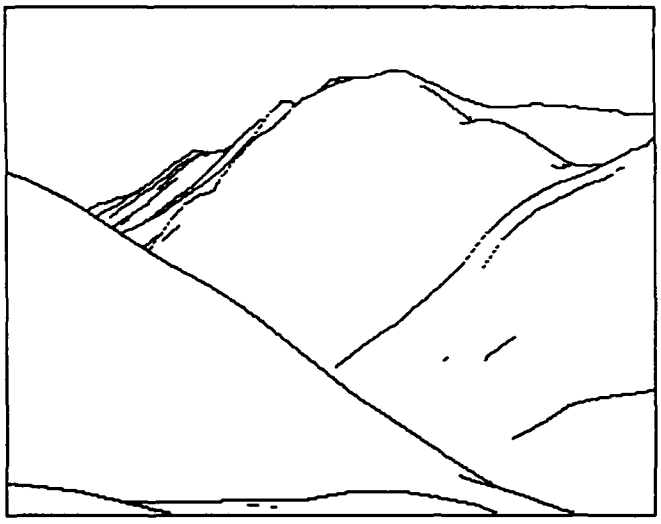


Figure 9: Ridgelines as determined from digital elevation data.

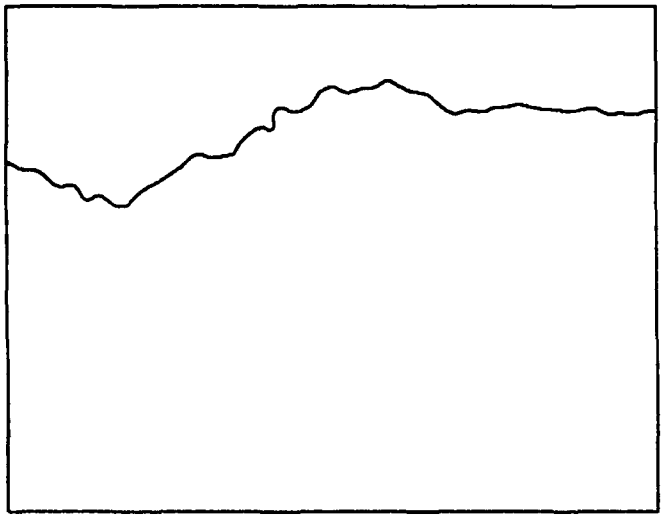


Figure 8: Extracted sky/terrain boundary.

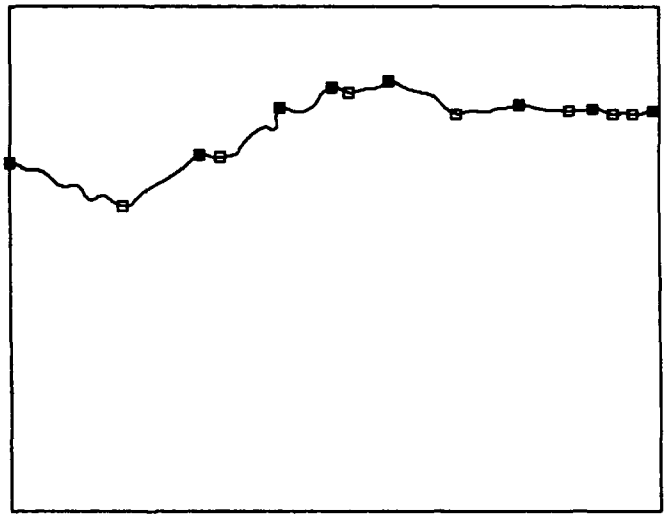


Figure 10: Labeling of topographic features.

[Mark, 1983] D. M. Mark. Automated detection of drainage networks from digital elevation models. In *Auto-Carto IV: Proceedings Sixth International Symposium on Computer Assisted Cartography*, pages 288–298, 1983.

[Martelli, 1972] A. Martelli. Edge detection using heuristic search methods. *Computer Graphics and Image Processing*, 1(2):169–182, 1972.

[Peucker and Douglas, 1975] T. K. Peucker and D. H. Douglas. Detection of surface-specific points by local parallel processing of discrete terrain elevation data. *Computer Graphics and Image Processing*, 4:375–387, 1975.

[Pick *et al.*, 1992] H. L. Pick, Jr., M. R. Heinrichs, D. R. Montello, K. Smith, C. N. Sullivan, and W. B. Thompson. Reading topographic maps. In J. Flach, [illegible] cock, J. Caird, and K. Vicente, editors, *Ecology of Human-Machine Systems*. 1992.

[Savitt, 1991] S. L. Savitt. *A Context Sensitive Segmenation Approach for Outdoor Terrain Feature Extraction*. PhD thesis, University of Minnesota, 1991.

[Speight, 1968] J. G. Speight. Parametric description of landforms. In G. A. Stewert, editor, *Land evaluation, Papers of a CSIRO Symposium on cooperation with UNESCO*, pages 239–250. 1968.

[Thompson *et al.*, 1990] W. B. Thompson, H. L. Pick, Jr., B. H. Bennett, M. R. Heinrichs, S. L. Savitt, and K. Smith. Map-based localization: The "drop-off" problem. In *Proc. DARPA Image Understanding Workshop*, pages 706–719, September 1990.

Colored Interreflections and Shape Recovery

Shree K. Nayar and Yitao Gong

Department of Computer Science, Columbia University, New York, N.Y. 10027

Abstract

Interreflections cause recovery algorithms to extract erroneous estimates of surface shape and color. Due to the complex nature of the interreflection phenomenon, its effects are difficult to analyze in a multi-dimensional color space. Noting that the interreflections of one wavelength are in general unaffected by those of any other wavelength, we decompose the interreflection process into three independent processes corresponding to the three spectral bands of color images. Photometric stereo is applied independently applied to each spectral band and recovered (pseudo) shapes and color are related to the actual shape and color of the surface. An algorithm is proposed that recovers the actual shape and color of the surface from the pseudo estimates. The accuracy and robustness of the algorithm is demonstrated on a variety of colored and multi-colored surfaces.

1 Interreflections and Shape Recovery

Points in the scene, when illuminated, reflect light not only in the direction of the sensor but also between themselves. This is always true with the exception of scenes that consist of only a single convex surface, in which case, no two points on the surface are visible to one another. In general, however, scenes concavities and points in the scene reflect light between themselves. These *interreflections* can appreciably alter the appearance of the scene. Existing vision algorithms do not account for the effects of interreflections and hence often produce erroneous results.

Two separate problems associated with interreflections can be identified; the *forward* (graphics) problem and the *inverse* (vision) problem. Most of the previous work done in this area is related to the forward problem. The forward problem involves the prediction of image brightness values given the shape and reflectance of a scene. Horn [Horn 70] discussed the changes in image intensities due to interreflections caused by polyhedral surfaces that are Lambertian in reflectance. Koenderink and van Doorn [Koenderink 83] formalized the interreflection process for Lambertian surfaces of arbitrary shape and varying reflectance (albedo). They proposed a solution to the forward problem in terms of the eigenfunctions of the interreflection kernel. Cohen and Greenberg [Cohen 85] modeled the scene as a finite collection of Lambertian planar facets and proposed a radiosity solution to the forward problem and used it to render images for graphics. Later, Forsyth and Zisserman [Forsyth 89] used a similar numerical solution to the forward problem to compare predicted and measured image intensities.

More recently, Nayar et.al [Nayar 91] demonstrated the effects of interreflections on shape-from-intensity algorithms, such as, shape-from-shading [Horn 70], photometric stereo [Woodham 78]. These are algorithms that recover three-dimensional shape information from image intensities. All shape-from-intensity methods, are based on the assumption that points in the scene are illuminated only by the sources of light and not other points in the scene; interreflections are assumed *not* to exist. As a result, these methods produce erroneous results when applied to concave surfaces. Nayar et.al analyzed the incorrect shape, the pseudo shape, recovered by shape-from-intensity methods when applied to concave Lambertian surfaces. They established a relation between the actual shape and pseudo shape and developed an algorithm that recovers the actual surface from the pseudo shape and reflectance estimates.

In their analysis, Nayar et.al [Nayar 91] focused on gray surfaces; each point on the Lambertian surface was assumed to have a constant albedo value that is independent of the wavelength of incident light. In the case of colored surfaces, however, the reflectance of a surface point is dependent on the spectral distribution of the incident light. In the case of concave surfaces, the spectral distribution of light rays incident on a given surface point depend on the spectral characteristics of both the source as well as other points on the surface. Further, the light reflected by the surface point also depends on its own spectral reflectance properties. Interreflections in the case of colored surfaces, therefore, can cause the color of one surface region to bleed onto another. These effects have been noted by other researchers [Bajcsy 89], [Brill 89], [Novak 90], and [Drew 90]. The common approach is to analyze these interreflection effects in a multi-dimensional color space. Since, interreflections are influenced by several factors including the unknown shape and color of the surface, the analysis in color space is difficult.

Figure 1 shows the effects of colored interreflections on photometric stereo. The surface has three regions of different color. The actual color of the surface points are seen to cluster at three points in the color space. The photometric

stereo method is applied independently to the three bands of the color image. As seen from the figure, three different shapes (all incorrect) are recovered from the three bands. The recovered color is also erroneous; it is not constant within each constant-color region of the surface.

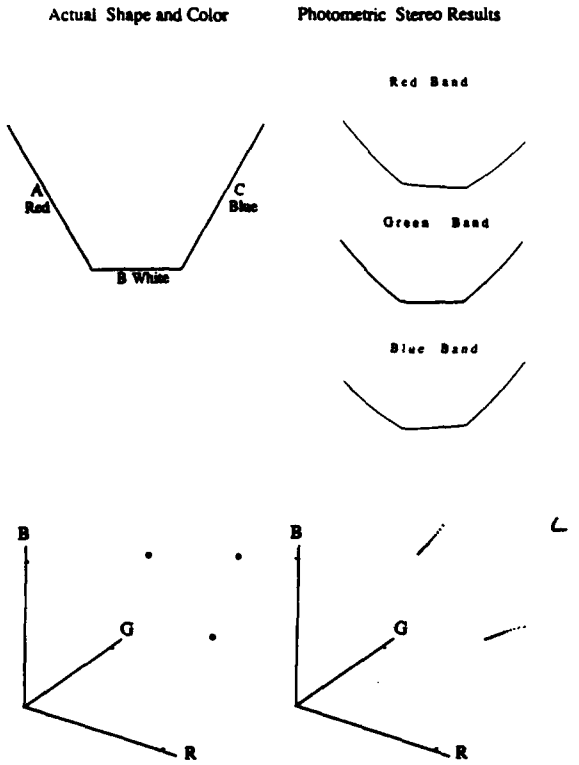


Figure 1: The actual shape and color of a surface and its shape and color computed by photometric stereo.

In this paper, we extend the results in [Nayar 91] to colored and multi-colored surfaces. Noting that the three bands of a color image correspond to differe. t wavelengths of light, we decompose the interreflection process into three independent processes. We then analyze the pseudo shape and pseudo albedo function produced by photometric stereo in each band. Then, we establish a relation between the pseudo shape and color and the actual shape and color of a surface. Using this relation, an algorithm is given that recovers the actual shape and color of the surface from the pseudo estimates. The algorithm is tested on a variety of multi-colored surfaces. The results demonstrate the robustness and accuracy of the algorithm.

2 A Diffuse Interreflection Model

Our solution to the inverse interreflection problem is based on the solution to the forward problem, i.e. modeling interreflections for diffuse surfaces of known shape and reflectance. The interreflection model described here is based on the formulation proposed by Koenderink and van Doorn [Koenderink 83]. All surfaces in the scene are assumed to be **Lambertian**. We will shortly see that this assumption is necessary to obtain a closed form solution to the forward interreflection problem. The Lambertian surface can have any arbitrary shape and varying reflectance, i.e. albedo (ρ) may vary from one surface point to the next. Here, we will assume that the incident light is **monochromatic**, i.e the surface reflects and interreflects light rays of a single wavelength. Hence each surface point can be assumed to have a constant reflectance coefficient (albedo). Later we show that the interreflection model developed here for monochromatic light can be extended and used for colored surfaces.

Consider the concave surface shown in Figure 2. The surface is divided into m infinitesimal facets. Let $\mathbf{x}_i$ and dx_i represent the three-dimensional coordinates and the surface area of the i^{th} facet, respectively. The radiance (brightness) and albedo values of each facet are assumed to be constant over the entire facet and equal to the radiance and albedo values at the center point $\mathbf{x}_i$ of the facet, i.e. $L_i = L(\mathbf{x}_i)$ and $\rho_i = \rho(\mathbf{x}_i)$. Consider the two facets i and j. The radiance of the facet i due to the radiance of the facet j is determined using basic radiometric definitions [Nicodemus 77] as:

$$L_i = \frac{\rho_i}{\pi} K_{ij} L_j \tag{1}$$

where the factor K_{ij} is given by:

$$K_{ij} = \frac{[\mathbf{n}_i \cdot \mathbf{r}_{ij}]\,[\mathbf{n}_j \cdot \mathbf{r}_{ji}]}{[\mathbf{r}_{ij} \cdot \mathbf{r}_{ij}]^2} \cdot dx_j \tag{2}$$

K_{ij} is a function of the relative positions and the orientations of the two facets; it determines the interreflections between i and j from a purely geometrical perspective. It is referred to as the interreflection **kernel**.

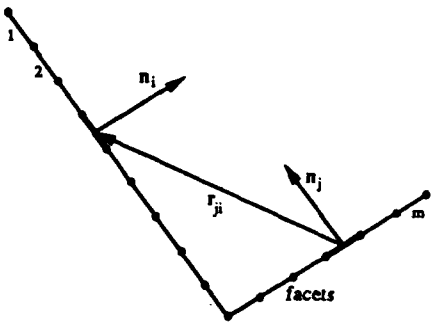


Figure 2: Modeling the surface as a collection of facets, each with its own radiance and albedo values.

Now, let us consider the entire surface shown in Figure 2. Assume the surface to be illuminated by a distant point source of light. Rays of light that impinge upon the surface are reflected between the facets. Since the albedo of

each facet is less than unity, the rays of light lose a fraction of their energy with each bounce. Eventually, after possibly an infinite number of bounces, the radiance at each point on the surface converges to a final steady-state value. Hence, the radiance of the facet i may be expressed as a sum of the radiance due to direct illumination from the source and the radiance due to the final radiance values of other facets on the surface:

$$L_i = L_{si} + \frac{\rho_i}{\pi} \sum_{j \neq i} L_j K_{ij} \quad (3)$$

where L_{si} is the radiance due to direct illumination from the source and the summation term corresponds to the radiance due to mutual illumination. This is the *interreflection equation*. Note that the radiance values L_i and L_j are assumed to be constants in the above equation. It is important to note that this assumption is valid only for Lambertian surfaces; the radiance of a Lambertian surface element is independent of the viewing direction.

The interreflection equation for the complete surface can be written using vector notation. We define the facet radiance vector as $\mathbf{L} = [\, L_1, L_2, ..., L_m \,]^T$ and the source contribution vector as $\mathbf{L_S} = [\, L_{s1}, L_{s2}, ..., L_{sm} \,]^T$. We also define the albedo matrix $\mathbf{P}$ and the kernel matrix $\mathbf{K}$ as:

$$\mathbf{P} = \frac{1}{\pi} \begin{bmatrix} \rho_1 & 0 & \dots & 0 \\ 0 & \rho_2 & 0 & \dots 0 \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \rho_m \end{bmatrix} \quad \mathbf{K} = \begin{bmatrix} 0 & K_{12} & \dots & \dots \\ K_{21} & 0 & \dots & \dots \\ \dots & \dots & 0 & \dots \\ \dots & \dots & \dots & 0 \dots \\ \dots & \dots & \dots & \dots 0 \end{bmatrix} \quad (4)$$

Then, equation 3 may be written as:

$$\mathbf{L} = \mathbf{L_S} + \mathbf{PKL} \quad (5)$$

or:

$$\mathbf{L} = (\mathbf{I} - \mathbf{PK})^{-1} \mathbf{L_S} \quad (6)$$

where $\mathbf{I}$ is the identity matrix. Thus, we have obtained a non-iterative, closed-form solution to the forward interreflection problem. The kernel and albedo matrices are determined by the shape and reflectance of the surface, respectively. The source direction and intensity may be used to compute the source contribution vector $\mathbf{L_S}$. Then the radiance of the surface facets, $\mathbf{L}$, can be determined using the above equation.

3 The Pseudo Shape

From equation 6 it is clear that surface radiance values are affected by interreflections. This indicates that if a shape-from-intensity method is applied to a concave surface it is expected to produce erroneous estimates of shape. In order to generalize the inverse interreflection problem, we assume that the reflectance of the Lambertian surface is also *unknown* and may vary from point to point. Therefore, by the term shape-from-intensity, we mean local methods that extract both shape (orientation) and reflectance (albedo) information. Photometric stereo [Woodham 78] is an example of such a shape-from-intensity method. In the presence of interreflections, photometric stereo extracts erroneous shape as well as erroneous reflectance estimates. We refer to the extracted shape as the pseudo shape and the extracted reflectance as the pseudo reflectance of the surface. In this section, we again assume that the light sources used to illuminate the surface are monochromatic. Under this assumption, we investigate how the pseudo shape and reflectance are related to the actual shape and reflectance of the surface. In the next section, we extend these results to colored interreflections.

Once again, consider the surface comprised of m facets (Figure 2). The i^{th} facet may be mathematically represented as:

$$\mathbf{N}_i = \frac{\rho_i}{\pi} \mathbf{n}_i \quad (7)$$

where $\mathbf{n}_i = [n_{xi}, n_{y_i}, n_{zi}]^T$ is the unit surface normal and ρ_i is the albedo value for the facet. Therefore, the term "facet" represents both local orientation as well as local reflectance information. The complete surface is then defined by the *facet matrix* $\mathbf{F} = [\, \mathbf{N}_1, \mathbf{N}_2, ..., \mathbf{N}_m \,]^T$. Consider, once again, the interreflection equation given by equation 6. Since the surface is Lambertian, the source contribution vector $\mathbf{L_S}$ may be determined from the facet matrix $\mathbf{F}$ and the source direction vector $\mathbf{s} = [s_x, s_y, s_z]^T$ as:

$$\mathbf{L_S} = \mathbf{F}\,\mathbf{s} \quad (8)$$

Hence, we obtain:

$$\mathbf{L} = (\mathbf{I} - \mathbf{PK})^{-1} \mathbf{F}\,\mathbf{s} \quad (9)$$

We define the matrix $\mathbf{F_p}$ as:

$$\mathbf{F_p} = (\mathbf{I} - \mathbf{PK})^{-1} \mathbf{F} \quad (10)$$

Note that $\mathbf{F_p}$ has the same dimensions as the facet matrix $\mathbf{F}$. In fact, in the absence of interreflections, $\mathbf{K}$ is a null matrix and $\mathbf{F_p} = \mathbf{F}$. In the presence of interreflections, $\mathbf{F_p}$ may be viewed as representing another Lambertian surface whose shape and reflectance differ from those of $\mathbf{F}$. Therefore, if photometric stereo is applied to the concave surface, the extracted shape and reflectance is $\mathbf{F_p}$ and not the actual shape and reflectance given by $\mathbf{F}$. We refer to $\mathbf{F_p}$ as the *pseudo facet matrix*; it represents the pseudo shape and pseudo reflectance that are extracted in the presence of interreflections.

In the case of photometric stereo, three different source directions, $\mathbf{s}_1$, $\mathbf{s}_2$, and $\mathbf{s}_3$, are used sequentially to illuminate the surface. The three resulting surface radiance vectors $\mathbf{L}_1$, $\mathbf{L}_2$, and $\mathbf{L}_3$ may be expressed as:

$$[\mathbf{L}_1, \mathbf{L}_2, \mathbf{L}_3] = \mathbf{F_p} . [\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3] \quad (11)$$

The pseudo facet matrix is computed as:

$$\mathbf{F}_p = [\mathbf{L}_1, \mathbf{L}_2, \mathbf{L}_3].[s_1, s_2, s_3]^{-1} \quad (12)$$

The i^{th} *pseudo facet* in $\mathbf{F}_p$ may be written as:

$$\mathbf{N}_{p_i} = \frac{\rho_{p_i}}{\pi} \mathbf{n}_{p_i} \quad (13)$$

where $\mathbf{n}_{p_i}$ and ρ_{p_i} are the pseudo surface normal and the pseudo albedo for the facet i and, in the presence of interreflections, differ from the actual surface normal and actual albedo of the facet.

We conclude this section by highlighting three important properties of the pseudo shape and reflectance:

- The pseudo shape and reflectance are *illumination invariant*. In equation 10, note that the albedo matrix $\mathbf{P}$, the kernel matrix $\mathbf{K}$, and the actual facet matrix $\mathbf{F}$ are all invariant to the direction and intensity of the illumination. As a result, the matrix $\mathbf{F}_p$ is also illumination invariant. It is independent of sources directions used by the shape-from-intensity method to illuminate the surface.

- The pseudo shape and reflectance are *unique*. From equation 10 we see that the pseudo facet matrix $\mathbf{F}_p$ is dependent on the actual facet matrix $\mathbf{F}$, the albedo matrix $\mathbf{P}$, and the kernel matrix $\mathbf{K}$. Note that $\mathbf{P}$ and $\mathbf{K}$ are in turn determined by $\mathbf{F}$. Hence, $\mathbf{F}_p$ is dependent only on $\mathbf{F}$. In other words, there exists only a single pseudo shape and pseudo reflectance corresponding to a given actual shape and reflectance.

- The pseudo shape tends to be *less concave* than the actual shape of the surface. A proof of this property is provided in [Nayar 90]. Figure 3 illustrates this property through a few examples of actual shapes and pseudo shapes All the surfaces are assumed to have a constant albedo value, $\rho = 0.95$. The pseudo shapes are computed using equation 10 and are seen to be less concave than the actual shapes. Figure 4 shows that the pseudo shape gets less concave as albedo increases.

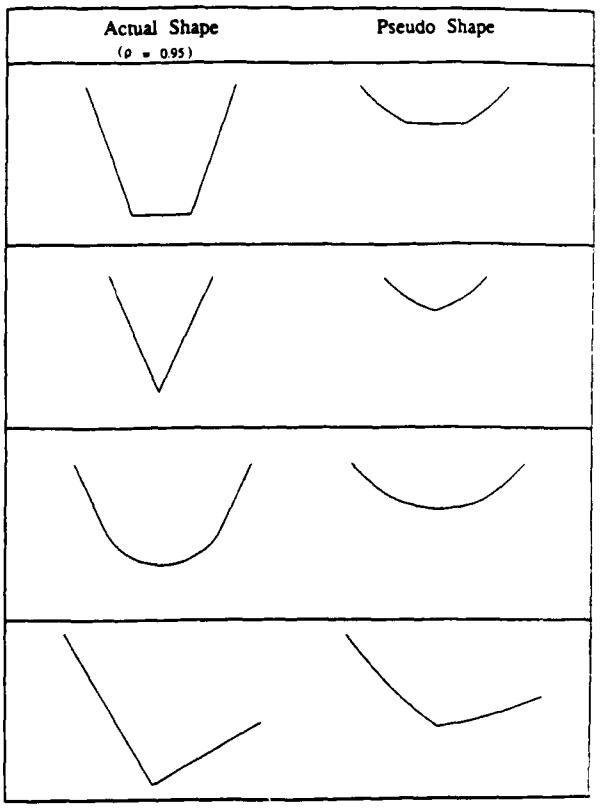


Figure 3: A few actual shapes (with $\rho = 0.95$) and their pseudo shapes.

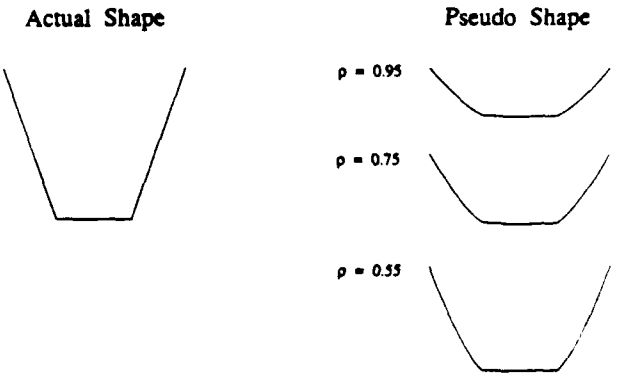


Figure 4: The difference between the actual and pseudo shapes increases with surface albedo.

4 Colored and Multi-Colored Surfaces

While developing the interreflection model (Section 2), we assumed a given surface facet has a constant albedo value. This assumption is only valid under either one of the following two conditions. (a) The surfaces are not colored but rather of different shades of gray and they reflect all wavelengths of incident light equally without attenuating some wavelengths more than others. (b) The incident light is monochromatic and therefore only light rays of a single wavelength are reflected and interreflected by the surface points. In the second case, since we are only concerned with a single wavelength, each point on the surface may be assumed to have a constant albedo value, namely, the albedo value for the given wavelength of incident light.

In the case of colored surfaces, the albedo of a facet would also depend on the spectral distribution of the incident light. In fact, with each bounce the spectral distribution of the reflected light would be altered depending on the spectral characteristics of the reflecting surfaces. Consider, for example, a concave surface that has two regions of different colors, say, red and blue. When the surface is illuminated, the red and blue regions interreflect incident light. If the surface is illuminated using white light, the blue region would reflect only blue light onto the red region. Therefore, a red surface point receives light rays of different spectral content; it receives white light from the source, blue light from the blue region, and red light from other red surface points. Further, the spectral distribution of light reflected by the red surface point depends not only on the spectral distribution of light received by it but also its own spectral characteristics (reflectance properties). As a result of the above effect, the color of a given surface region may be altered due to reflections from neighboring regions that have different colors. This phenomenon is often referred to as *color bleeding*. The reader is directed to [Bajcsy 89], [Brill 89], [Novak 90], and [Drew 90] for a more detailed discussion on this effect.

In the case of colored surfaces, therefore, our assumption that each surface point has a constant albedo value is no longer valid: The albedo of a point would depend on its color and the spectral distribution of the incident light, where incident light includes light received from other surface points. However, from condition (b) we know that, for a given wavelength, a surface point (irrespective of its color) can be assumed to have a unique albedo value. Therefore, pseudo shape and reflectance estimates for a multi-colored surface can be computed by using a narrow-band filter at the sensor end so that only light waves of a particular wavelength are detected by the sensor. Note that the pseudo shape and reflectance will in general be different for different wavelengths.

Typically, color images are obtained using three narrow-band filters at the sensor end. These three filters have narrow-band spectral responses in the red, green, and blue regions of the visible light spectrum. We will refer to the three images produced as the red, green, and blue bands of the color image. Since the three filters are narrow-band filters, the image produced by each filter represents reflections and interreflections of almost a single wavelength of light. Hence, a pseudo shape and pseudo albedo function for the surface can be computed using each band of the color image. The pseudo shapes and albedo functions for the three band are expected to be different since each point may have a different actual albedo for the three different wavelengths of light. Hence, the pseudo facet matrices for the three band may be written as:

$$\begin{aligned} \mathbf{F}_p{}^{\mathrm{R}} &= (\mathbf{I} - \mathbf{P}^{\mathrm{R}}\mathbf{K})^{-1}\mathbf{F}^{\mathrm{R}} \\ \mathbf{F}_p{}^{\mathrm{G}} &= (\mathbf{I} - \mathbf{P}^{\mathrm{G}}\mathbf{K})^{-1}\mathbf{F}^{\mathrm{G}} \\ \mathbf{F}_p{}^{\mathrm{B}} &= (\mathbf{I} - \mathbf{P}^{\mathrm{B}}\mathbf{K})^{-1}\mathbf{F}^{\mathrm{B}} \end{aligned} \quad (14)$$

where $\mathbf{P}^{\mathrm{R}}$, $\mathbf{P}^{\mathrm{G}}$, and $\mathbf{P}^{\mathrm{B}}$ are the albedo matrices of the surface for three wavelengths determined by the three filters. Note that the interreflection kernel $\mathbf{K}$ is a purely geometrical quantity and hence remains the same for all wavelengths. $\mathbf{F}^{\mathrm{R}}$, $\mathbf{F}^{\mathrm{G}}$, and $\mathbf{F}^{\mathrm{B}}$ represent the actual shape and color of the surface.

The three pseudo shapes and the pseudo color of the surface may be estimated using photometric stereo. Let the three sources used by photometric stereo have directions s_1, s_2, and s_3. We assume that all three sources emit white light and hence have the same radiant intensity for the three wavelengths. Then, the pseudo facet matrices for the different color bands are determined as:

$$\begin{aligned} \mathbf{F}_p{}^{\mathrm{R}} &= [\mathbf{L}_1{}^{\mathrm{R}}, \mathbf{L}_2{}^{\mathrm{R}}, \mathbf{L}_3{}^{\mathrm{R}}].[s_1, s_2, s_3]^{-1} \\ \mathbf{F}_p{}^{\mathrm{G}} &= [\mathbf{L}_1{}^{\mathrm{G}}, \mathbf{L}_2{}^{\mathrm{G}}, \mathbf{L}_3{}^{\mathrm{G}}].[s_1, s_2, s_3]^{-1} \\ \mathbf{F}_p{}^{\mathrm{B}} &= [\mathbf{L}_1{}^{\mathrm{B}}, \mathbf{L}_2{}^{\mathrm{B}}, \mathbf{L}_3{}^{\mathrm{B}}].[s_1, s_2, s_3]^{-1} \end{aligned} \quad (15)$$

The above equations indicate that the same surface produces three different pseudo shapes and albedo functions. By noting that the three bands of a color image correspond to different wavelengths of light, we have been able to decompose the interreflection process into three independent processes. Having done this we are in a position to analyze the interreflections in each band independent of the other bands.

5 Recovering Actual Shape and Color

In [Nayar 91] an algorithm is developed that uses equation. 10 to iteratively recover the actual shape and reflectance of a gray surface from its pseudo shape and reflectance. Here, we briefly describe the algorithm for gray surfaces and then extend it to colored surfaces. At first, photometric stereo is applied to the scene. If the scene consists of a single convex surface, the extracted pseudo shape and reflectance are simply the actual ones. However, if the scene consists of concavities, the pseudo shape and reflectance differ from the actual ones. As we showed in Section 3, the pseudo shape is a shallower (less concave) version of the actual shape. Hence, the algorithm uses the pseudo shape and reflectance as conservative initial estimates of the actual shape and reflectance, to compute initial estimates for the albedo matrix $\mathbf{P}$ and the kernel matrix $\mathbf{K}$. The computed $\mathbf{P}$, $\mathbf{K}$, and the pseudo facets $\mathbf{F}_p$ are then inserted in equation 10 to obtain the next estimate of the actual facets. This estimate of the surface is expected to be more concave than the previous one and is used in the next iteration to obtain an even better estimate. The algorithm may hence be written as:

$$\begin{aligned} \mathbf{F}^{k+1} &= (\mathbf{I} - \mathbf{P}^k\mathbf{K}^k)\mathbf{F}_p \\ \text{where } \mathbf{F}^0 &= \mathbf{F}_p \end{aligned} \quad (16)$$

In the above equation, $\mathbf{P}^k = \mathbf{P}(\mathbf{F}^k)$ and $\mathbf{K}^k = \mathbf{K}(\mathbf{F}^k)$. Note that each estimate of $\mathbf{F}$ provides estimates of *both* shape and reflectance. With each iteration, more accurate estimates of shape and reflectance are obtained and the result finally converges to the actual shape and reflectance.

In implementing the algorithm, the surface is assumed to be continuous. The interreflection kernel depends not only on the orientations of individual facets but also on their relative positions. Therefore, a depth map of the scene must be reconstructed (by integration) from the orientation map computed in each iteration of the algorithm. The continuity assumption is necessary to ensure integrability of the orientation maps.

Several experiments were conducted on real surfaces and the results indicate that the algorithm is robust and accurate [Nayar 90]. The convergence properties of the algorithm are discussed in detail in [Nayar 90].

We now extend the algorithm to recover true shapes and colors of colored surfaces. First, equation 15 is used to estimate the three pseudo shapes and the pseudo color of the surface using photometric stereo. Then, the actual shape and color of the surface are iteratively recovered from the pseudo estimates. Here, the recovery algorithm given by equation 16 is applied to each of the three pseudo facet matrices $\mathbf{F}_p{}^{\mathrm{R}}$, $\mathbf{F}_p{}^{\mathrm{G}}$, and $\mathbf{F}_p{}^{\mathrm{B}}$. The recovery algorithm for colored surfaces may be written as:

$$\mathbf{F}^{\mathrm{R}\,k+1} = (\mathbf{I} - \mathbf{P}^{\mathrm{R}\,k}\mathbf{K}^k)\,\mathbf{F}_p{}^{\mathrm{R}} \quad (17)$$

$$\mathbf{F}^{\mathrm{G}\,k+1} = (\mathbf{I} - \mathbf{P}^{\mathrm{G}\,k}\mathbf{K}^k)\,\mathbf{F}_p{}^{\mathrm{G}} \quad (18)$$

$$\mathbf{F}^{\mathrm{B}\,k+1} = (\mathbf{I} - \mathbf{P}^{\mathrm{B}\,k}\mathbf{K}^k)\,\mathbf{F}_p{}^{\mathrm{B}} \quad (19)$$

where:

$$\mathbf{F}^{\mathrm{R}\,0} = \mathbf{F}_p{}^{\mathrm{R}} \quad (20)$$

$$\mathbf{F}^{\mathrm{G}\,0} = \mathbf{F}_p{}^{\mathrm{G}}$$

$$\mathbf{F}^{\mathrm{B}\,0} = \mathbf{F}_p{}^{\mathrm{B}}$$

Note that the recovery algorithm is applied independently to the three pseudo facet matrices $\mathbf{F}_p{}^{\mathrm{R}}$, $\mathbf{F}_p{}^{\mathrm{G}}$, and $\mathbf{F}_p{}^{\mathrm{B}}$ to recover the facet matrices $\mathbf{F}^{\mathrm{R}}$, $\mathbf{F}^{\mathrm{G}}$, and $\mathbf{F}^{\mathrm{B}}$, respectively. Therefore, we obtain three estimates of the actual shape of the surface, one obtained from each one of the three pseudo facet matrices. The three shape estimates are identical and correspond to the true shape of the colored surface. Therefore, the shape of the surface may be recovered from any one of the three pseudo facet matrices. However, in order to recover the actual color at each point of the surface, we need to recover all three facet matrices.

The above algorithm is applicable to multi-colored surfaces, i.e. surfaces that are comprised of regions of different colors. Once again, the surface is assumed to be continuous to ensure integrability of orientation maps produced in each iteration. We also assume that each surface point has a non-zero albedo value for all three wavelengths of the color image. This assumption is not a severe one as most surfaces encountered in practice have non-zero albedo values for all wavelengths in the visible spectrum.

6 Results

The recovery algorithm has been tested on a variety of gray surfaces and the results indicate that the algorithm is robust and accurate in recovering the actual shape and reflectance of gray surfaces [Nayar 90]. Here, we present the simulation results of applying the algorithm to colored surfaces. The surfaces are three-dimensional with translational symmetry in one direction. The interreflection kernel for the transalational symmetry case was derived by Forsyth and Zissermann [Forsyth 89] and is given in Appendix A.1.

The example shown in Figure 5 is a bucket shaped surface that has three faces with different colors, namely, red, white, and blue. The red face of the surface reflects more red light than any other wavelength in the visible spectrum. Note that the red region does reflect green and blue light but in less amounts relative to red light. We assume that the color images are obtained using three narrow-band filters that pass red light, green light, and blue light. Each color image therefore has three bands, namely, the red band, green band, and the blue band. The actual color of the surface is shown by plotting the three albedo functions (along the surface crossection) for the three wavelengths (red, green, and blue). Within each of the three regions (A, B, and C) on the surface, the albedo functions are constant since all surface points within a region have the same color.

Photometric stereo is used to compute a pseudo shape for each band of the color image. The three pseudo shapes are shown in Figure 5. Note that the pseudo shapes produced by the three bands are different. The pseudo shape and albedo produced by the green band is symmetric since the actual shape and the actual albdeo function for green light are symmetric. On the other hand, the pseudo shapes and albedo functions for the red and blue bands are asymmetric since the actual albedo function of the surface for red and blue light are asymmetric. The pseudo albedo values are often greater that the actual albedo values, sometimes exceeding unity. This results from the fact that interreflections always increase the radiance of a surface point.

At the bottom of Figure 5 we show the results of applying the recovery algorithm given by equation 17. For all three bands the algorithm successfully recovers the actual shape and albedo of the surface. Note that the algorithm must be applied to all three bands to recover the true color of the surface. To recover the shape, however, the algorithm need be applied only to any one of the three band.

Figures 6 and 7 show results obtained for other multicolored surfaces. In all the results included here and numerous other unreported results, the algorithm convergence (with near zero error) in about 8 iterations of the algorithm.

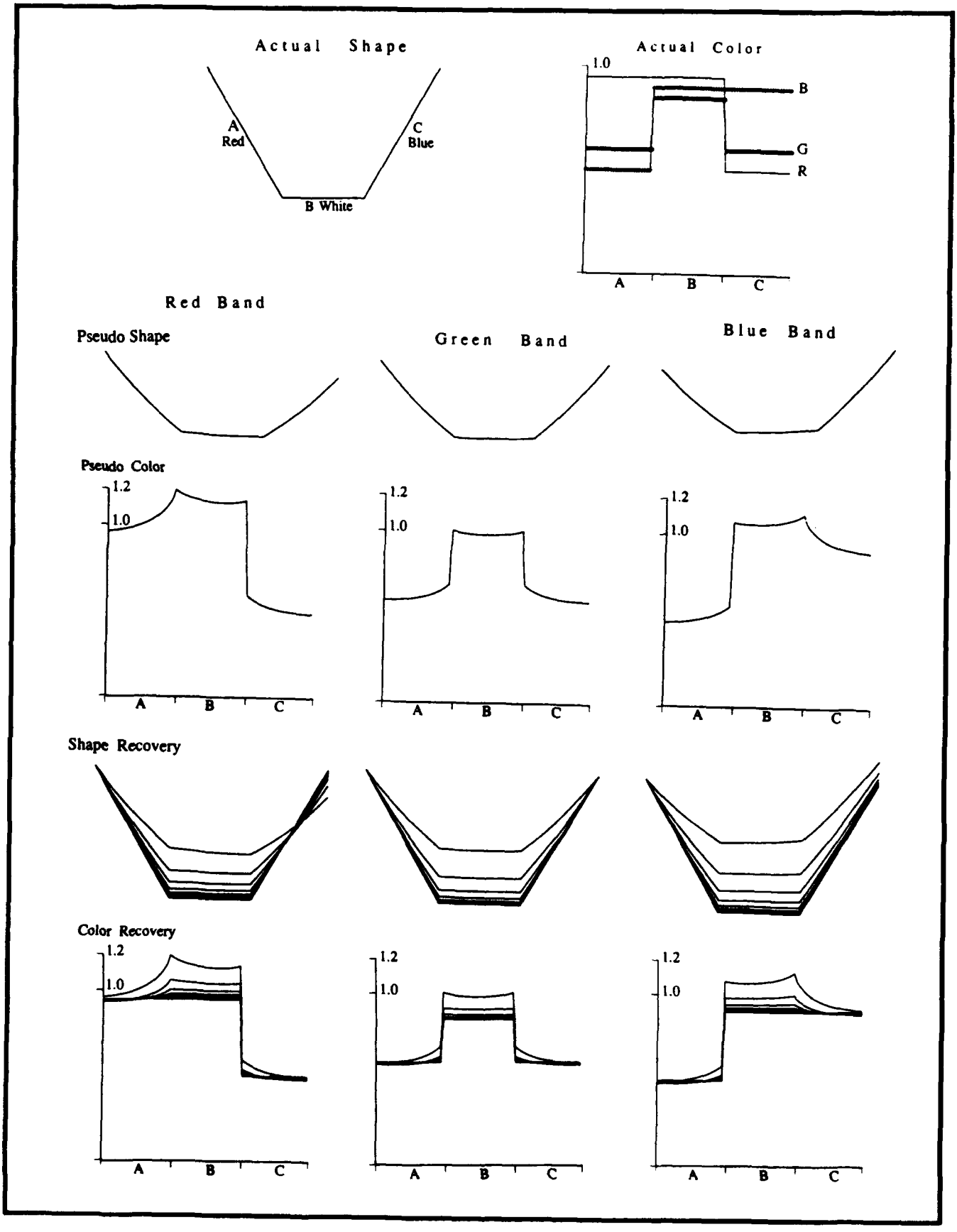


Figure 5: Results: Multi-colored surface with a bucket shaped cross-section.

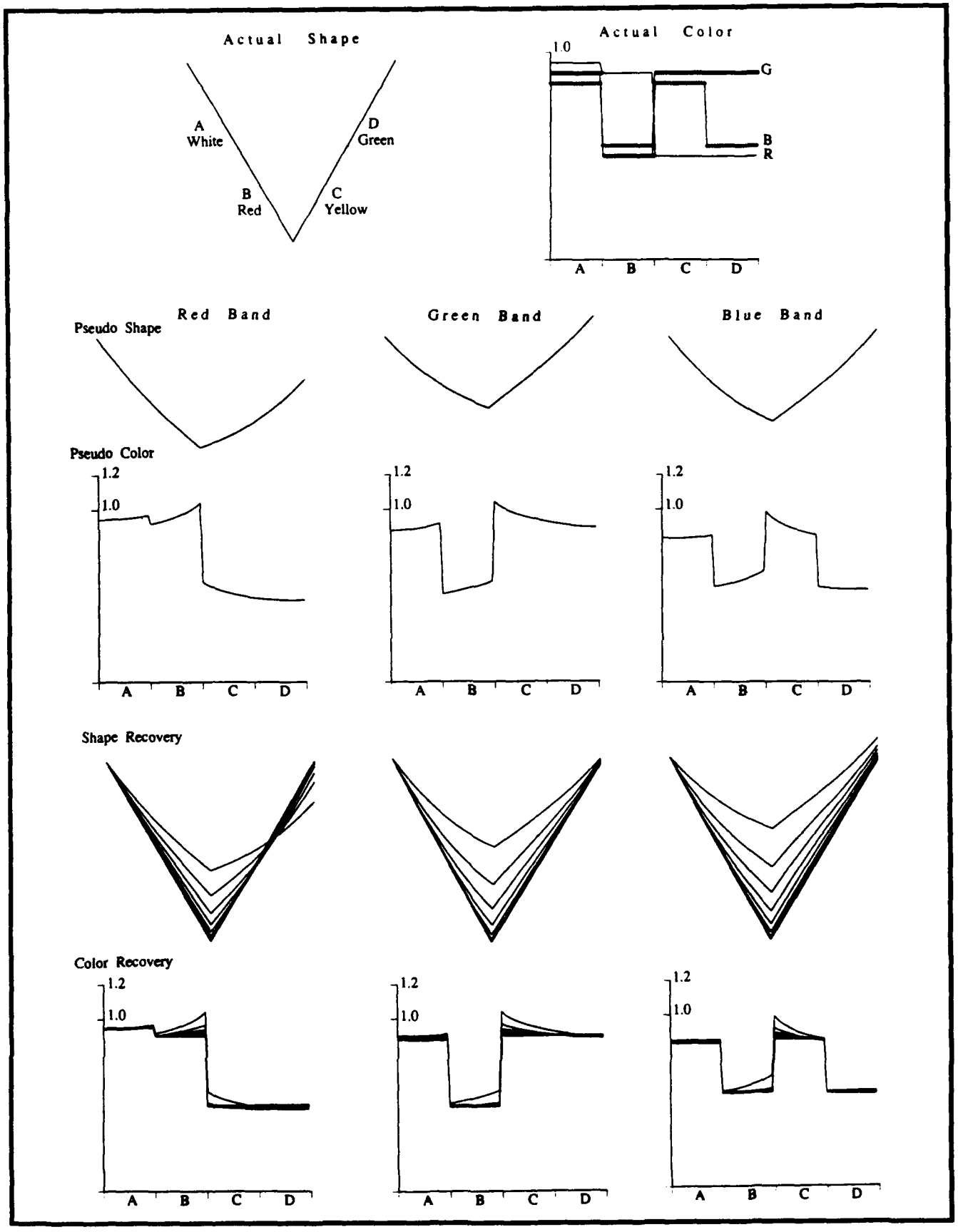


Figure 6: Results: Multi-colored surface with a V-groove cross-section.

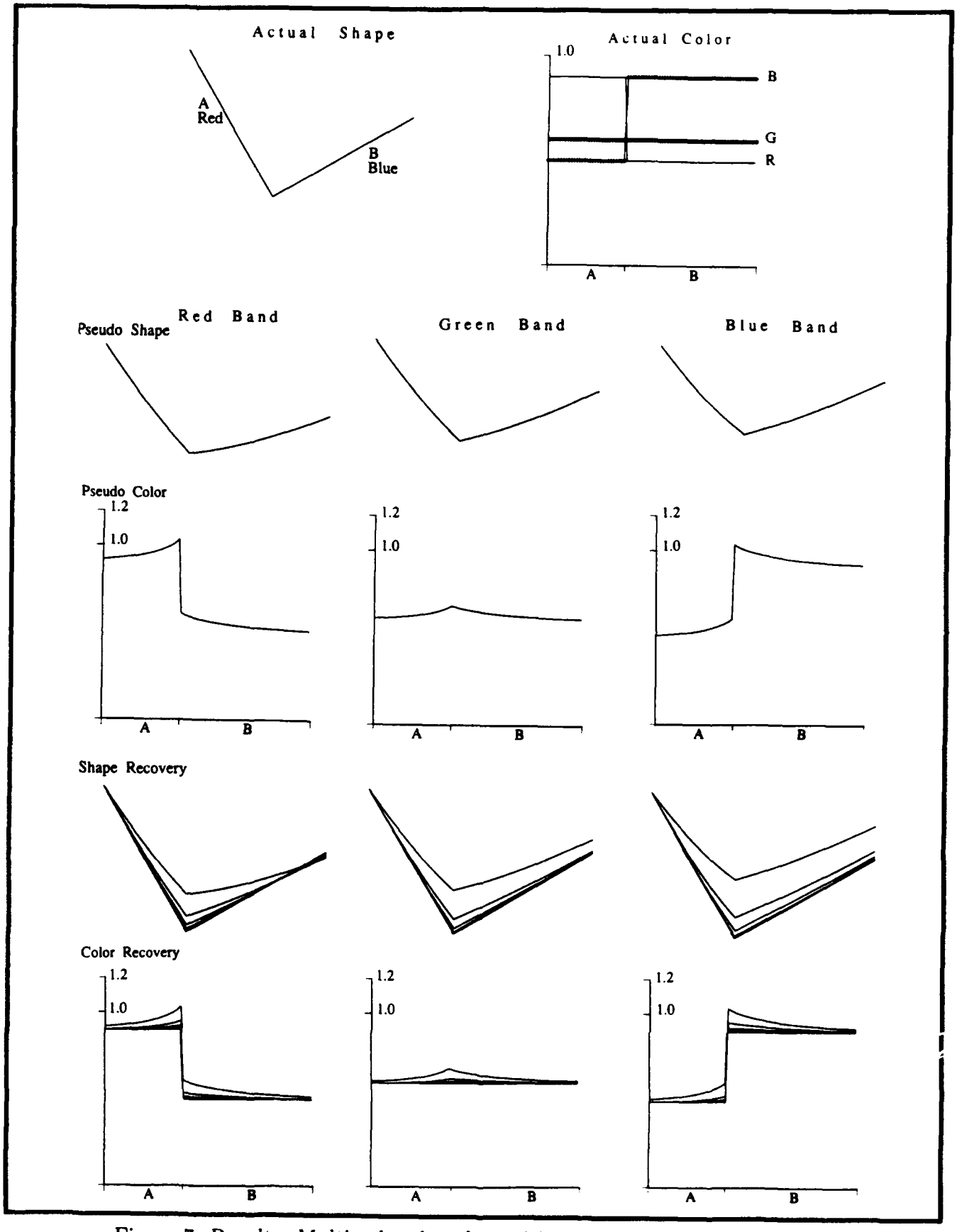


Figure 7: Results: Multi-colored surface with a titled V-groove cross-section.

7 Conclusion

We have presented an algorithm for recovering the shape and reflectance of colored surfaces in the presence of diffuse interreflections. The surfaces may be of arbitrary but continuous shape, and with possibly varying and unknown color.

We showed that for a Lambertian surface the shape and color extracted by shape-from-intensity methods are incorrect in the presence of interreflections. Interreflections can cause the color of one region on a surface to bleed onto a neighboring region. In general these effects are difficult to analyze in a multi-dimensional color space. Noting that the three different bands of a color image correspond to different wavelengths of light, we decomposed the interreflection process into three independent processes. We then showed that the pseudo shape and color extracted using photometric stereo, though incorrect, can be related to the actual shape and color of the surface. The pseudo estimates are shown to have interesting invariance properties; for any given Lambertian surface there exists a single pseudo shape and pseudo color that are invariant to the source directions used to recover them.

Using the properties of the pseudo surface, we developed an algorithm that recovers the actual shape and color of the surface from the pseudo estimates. The algorithm was applied to a variety of multi-colored test surfaces and was shown to be robust and accurate. Motivated by the results presented in this paper, we are currently conducting experiments on real multi-colored surfaces.

A Appendix

A.1 Kernel for Translational Symmetry Case

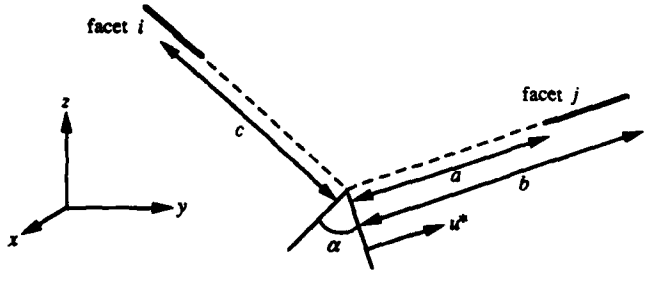


Figure 8: Cross-sectional view of two planar facets that are infinite in the x direction.

Forsyth and Zisserman [Forsyth 89] have derived the interreflection kernel for the special case of two planar facets that have translational symmetry in a single direction. Figure 8 shows a cross-sectional view of two such facets that are infinite in the x direction. The kernel K_{ij} is derived [Forsyth 89] by integrating along the x and y directions, the contribution of all points on facet j to the radiance of a point on the facet i:

$$K_{ij} = -\frac{\pi}{2}\left[\frac{c + u^* cos\alpha}{\left(c^2 + 2\,c\,u^* cos\alpha + u^{*2}\right)^{1/2}}\right]_{u^*=a}^{u^*=b} \quad (21)$$

where α is the angle between the surface normal vectors of the two facets, and the parameter u^* represents the cross-sectional length of the facet j. Since both facets are infinite in length, the same kernel is valid for all points on the facet i. Therefore, under the translational symmetry assumption, the kernel need only be evaluated for points along the cross-section of the surface. Note that the above kernel is valid only for surfaces that are infinite in the direction of symmetry. However, the kernel serves as a good approximation [Forsyth 89] for points that lie around the middle of a surface that is long though finite in the direction of symmetry.

Acknowledgements

The authors are grateful to Ushir Shah for his help in preparing the final manuscript.

References

[Bajcsy 89] R. Bajcsy, S. W. Lee, and A. Leonardis "Image Segmentation with Detection of Highlights and Interreflections Using Color," GRASP LAB 182 MS-CIS-89-39, University of Pennsylvania, Dept. of Computer and Info. Science, June 1989.

[Blinn 77] J. F. Blinn, "Models of Light Reflection for Computer Synthesized Pictures," *ACM Computer Graphics*, Vol. 11, No. 2, pp. 192-198, July, 1977.

[Brill 89] M. H. Brill, "Object-based segmentation and color recognition in multispectral images," *SPIE-SPSE Meeting*, January, 1989, Paper 1076-11.

[Cohen 85] M. F. Cohen and D. P. Greenberg, "The Hemicube: A Radiosity Solution for Complex Environments," *SIGGRAPH 1985*, 19, pp. 31-40, 1985.

[Drew 90] M. S. Drew and B. V. Funt, "Calculating Surface Reflectance using a Single-Bounce Model of Mutual Reflection," *Proc. Third International Conference on Computer Vision*, pp. 394-399, December, 1990.

[Forsyth 89] D. Forsyth and A. Zisserman, "Mutual Illumination," *Proc. of CVPR*, pp. 466-473, 1989.

[Forsyth 90] D. Forsyth and A. Zisserman, "Shape from shading in the light of mutual illumination," *Image and Vision Computing*, Vol. 8, No. 1, pp. 42-49, February 1990.

[Gilchrist 79] A. L. Gilchrist, "The Perception of Surface Blacks and Whites," *Scientific American*, bf 240, pp. 112-124, 1979.

[Horn 70] B. K. P. Horn, "Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View," MIT Project MAC Internal Report TR-79 and MIT AI Laboratory Technical Report 232, November, 1970.

[Horn 77] B. K. P. Horn, "Image intensity understanding," *Artificial Intelligence*, Vol. 8, No. 2, pp. 201-231, 1977.

[Horn 89] B. K. P. Horn and M. J. Brooks, editors, *Shape from Shading*, MIT Press, 1989.

[Jacquez 55] J. A. Jacquez and H. F. Kuppenheim, "Theory of the integrating sphere," *Journal of Optical Society of America*, Vol. 45, pp. 460-470, 1955.

[Koenderink 83] J. J. Koenderink and A. J. van Doorn, "Geometrical modes as a general method to treat diffuse interreflections in radiometry," *Journal of Optical Society of America*, Vol. 73, No. 6, pp. 843-850, June, 1983.

[Nayar 90] S. K. Nayar, *Shape Recovery using Physical Models of Reflection and Interreflection*, Ph.D. Dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University, December, 1990.

[Nayar 91] S. K. Nayar, K. Ikeuchi, T. Kanade, "Shape from Interreflections," *International Journal of Computer Vision*, Vol. 6, No. 3, pp. 173-195, August, 1991.

[Nicodemus 77] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis, "Geometrical Considerations and Nomenclature for Reflectance," NBS Monograph 160, National Bureau of Standards, October 1977.

[Novak 90] C. Novak, "Ph.D Thesis Proposal," Department of Computer Science, Carnegie Mellon University, August 1990.

[Woodham 78] R. J. Woodham, "Photometric stereo: A reflectance map technique for determining surface orientation from image intensity," *Proc. SPIE*, Vol. 155, pp. 136-143, 1978.

Consensus by Decomposition: A Paradigm for Fast High Breakdown Point Robust Estimation*

Doron Mintz
LSI Logic Corp.
1501 McCarthy Blvd.
Milpitas, CA 95035

Peter Meer
Department of Electrical and Computer Engineering
Rutgers University
Piscataway, NJ 08855-0909

Azriel Rosenfeld
Center for Automation Research
University of Maryland
College Park, MD 20742

Abstract

Robust high breakdown point techniques are gaining increasing popularity in computer vision since they can tolerate up to half the data being severely corrupted. The least median of squares (LMedS) estimator is the best known example of such techniques. We show that the attractive properties of the LMedS estimator do not hold when all the data is corrupted by zero mean noise. We propose a new "consensus by decomposition" (CBD) algorithm which preserves the properties of LMedS up to low signal-to-noise ratios while achieving a significant speed-up relative to LMedS. The CBD estimator uses a different paradigm than LMedS. The data is decomposed in both the spatial and parameter domains. A separate distribution is built for every parameter and the distribution is analyzed with a new, enhanced mode detection procedure. The superiority of the CBD estimator is proved by extensive simulations.

1 Introduction

The riddle of the "chicken and egg" is a well known teaser. Which one was first? Since each one has a causality relation with the other neither of them can be chosen as primal. In computer vision similar situations are often met. For example, consider the problem of segmentation. To segment an image we must delineate regions homogeneous under a given measure. The boundaries of the regions are the discontinuities in the homogeneity measure. Most of the methods of detecting homogeneity, however, cannot handle data containing discontinuities. To avoid erroneous results we would like to analyze the data only far away from discontinuities. But the locations of the discontinuities are not available since this is one of the reasons why we want to segment the image! See [6] for a discussion of the segmentation problem.

We conclude that the "chicken and egg" problem is caused by the inability of our methods to handle piecewise data, i.e., data containing discontinuities. The weighted least squares (WLS) estimator is often a basic computational module of such methods. The limitations of the WLS estimator and its properties that are relevant to computer vision are discussed in Section 2.

In the last two decades, to avoid the pitfalls of least squares, robust estimation techniques were developed by statisticians. Recently these robust techniques have become popular in computer vision. See [5] for a survey of robust techniques as applied in computer vision. The class of high breakdown point robust operators can eliminate the above mentioned "chicken and egg" situation. These operators can tolerate a discontinuity and recover the model corresponding to the absolute majority of the data. The least median of squares (LMedS) estimator, proposed by Rousseeuw in 1984, is the most frequently used in computer vision. However, the LMedS estimator was developed for statistical applications and its utility in solving computer vision problems cannot be taken for granted. In Section 3 we describe LMedS and show its limitations for noisy data.

In Section 4 we propose a new high breakdown robust algorithm which we call "consensus by decomposition" (CBD). The CBD estimator retains the desirable properties of both the least squares and the least median of squares estimators while eliminating most of their drawbacks. The algorithm was developed for computer vision applications and its characteristics are analyzed through simulation studies. Some of the questions raised by the CBD estimator, its relation to the other robust techniques used in computer vision, and the consensus vision paradigm are discussed in Section 6.

2 The Weighted Least Squares Estimator

The weighted least squares (WLS) estimator is a frequently used tool in computer vision. We only mention here the properties used later in the paper. For proofs and/or more detailed discussions see a textbook on linear estimation. Let $z_{1,1} \dots z_{n,m}$ be the data and assume that it can be modeled by a *linear* model characterized by p parameters β_k, $k = 0 \dots (p-1)$. Thus

$$z_{i,j} = \sum_{k=0}^{p-1} \beta_k x_k(i,j) + v_{i,j} \qquad (1)$$

where the p regressor variables $x_k(i,j)$ are specified for every sampling site (i,j), $i = 0 \dots n$; $j = 0 \dots m$; and

*The support of the Air Force Office of Scientific Research under Grant AFOSR-86-0092 is gratefully acknowledged.

$v_{i,j}$ represents the deviation from the model, i.e., the noise. The problem of non-linear models and of uncertainty about the regressor variables is beyond the scope of this paper.

2.1 Properties of WLS

In matrix notation (1) becomes

$$\mathbf{z} = \mathbf{X}\boldsymbol{\beta} + \mathbf{v} \tag{2}$$

where $\mathbf{z}$ is a $nm \times 1$ vector of data, $\mathbf{X}$ is a $nm \times p$ matrix of regressor variables, $\boldsymbol{\beta}$ is the unknown $p \times 1$ vector of regression coefficients, and $\mathbf{v}$ is a $nm \times 1$ noise vector. To find the weighted least squares estimate of the parameter vector, $\hat{\boldsymbol{\beta}}$, we must minimize the quadratic form

$$(\mathbf{z} - \mathbf{X}\hat{\boldsymbol{\beta}})^t \mathbf{W} (\mathbf{z} - \mathbf{X}\hat{\boldsymbol{\beta}}) \tag{3}$$

where $\mathbf{W}$ is a symmetric, positive definite $nm \times nm$ weight matrix, and the superscript t means transpose.

The minimization of (3) yields a reliable estimate only if the noise has zero mean, i.e., $\mathrm{E}[\mathbf{v}] = 0$. ($\mathrm{E}[\cdot]$ is the expectation operator.) The noise, however, can be correlated, having the covariance matrix $\mathrm{cov}[\mathbf{v}] = \mathbf{V}$. The weighted least squares estimate is obtained by computing the expression

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^t \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^t \mathbf{W} \mathbf{z}. \tag{4}$$

If the noise has zero mean the estimate (4) is unbiased:

$$\mathrm{E}\left[\hat{\boldsymbol{\beta}}\right] = \boldsymbol{\beta}. \tag{5}$$

This very important property is satisfied for any weight matrix $\mathbf{W}$. If we choose $\mathbf{W} = \mathbf{V}^{-1}$, $\hat{\boldsymbol{\beta}}$ is also the minimum-variance estimate among all unbiased estimates linearly related to the measurements. In this case

$$\mathrm{cov}\left[\hat{\boldsymbol{\beta}}\right] = (\mathbf{X}^t \mathbf{V}^{-1} \mathbf{X})^{-1} \tag{6}$$

If in addition $\mathbf{v}$ has a multivariate normal distribution, $\hat{\boldsymbol{\beta}}$ is also the maximum likelihood estimate.

2.2 WLS in Computer Vision

Minimum-variance estimation, however, may not suffice in computer vision applications in which a local operator deals with small sample sizes. For example, let us estimate the coefficients of a planar fit ($p = 3$) in the presence of i.i.d. noise ($\mathrm{cov}[\mathbf{v}] = \sigma^2\mathbf{I}$), for data defined in an odd-sized square window on a dense sampling grid. The model is then of the form

$$\hat{z}_{i,j} = \hat{\beta}_0 + \hat{\beta}_1 i + \hat{\beta}_2 j \qquad i, j = -n \ldots n. \tag{7}$$

Since we have assumed i.i.d. noise we must perform unweighted least squares for minimum-variance estimation, and it is immediate to obtain from (6)

$$\mathrm{Cov}\left[\hat{\boldsymbol{\beta}}\right] = \sigma^2 \begin{pmatrix} \frac{1}{(2n+1)^2} & 0 & 0 \\ 0 & \frac{3}{n(n+1)(2n+1)^2} & 0 \\ 0 & 0 & \frac{3}{n(n+1)(2n+1)^2} \end{pmatrix}. \tag{8}$$

The covariance matrix is diagonal and thus the estimates of the three parameters, the intercept and the two slopes, are uncorrelated. This is due to the fact that the column vectors of $\mathbf{X}$ are orthogonal. It is important to notice that for a higher degree polynomial surface model, or for data defined on a sparse grid, the column vectors are not necessarily orthogonal and some of the estimate vector components may become correlated.

The estimate is unbiased and we can use as a simple quality measure the ratio between the correct parameter value β_k and the standard deviation of its estimate taken from (8). We will consider an estimate reliable if the ratio is larger than 5. (The more accurate measure of a confidence interval involves additional assumptions about the noise.) Thus we can obtain the normalized upper bound for the noise standard deviation yielding reliable slope estimates ($k = 1$ or 2). In Table 1 the values of this bound are shown for several window sizes.

Table 1: Dependence of the normalized noise upper bound on window size.

n	Window Size	Upper Bound
1	3×3	0.49
2	5×5	1.41
3	7×7	2.80
4	9×9	4.65

In images the slope values have the order of magnitude of units. For example if the plane is slanted at 45 degrees ($\beta_1 = 1$) a 5×5 window is able to handle only $\sigma \leq 1.41$, not far from the range of quantization noise. Similar bounds can be obtained for the coefficients of higher degree polynomial surfaces. We conclude that the small windows usually employed in computer vision for local estimation of polynomial surfaces cannot return reliable parameter estimates even in the presence of negligible noise. Only the intercept estimate is reliable, a fact used in image smoothing operators applied at every pixel.

The implicit assumption behind the model (1) is that the entire data set can be characterized by only *one* parameter vector $\boldsymbol{\beta}$. For piecewise data, a case frequently met in computer vision but less common in statistics, this is not true. In piecewise data a model discontinuity is also present and (at least) two parameter vectors are required to describe the entire data set. For example, changes in reflectance or depth yield piecewise data.

It is well known that least squares estimation cannot handle piecewise data. This is caused by the violation of the zero mean noise assumption. Indeed, if we assume the validity of (1) we must take one of the models as fitting all the data. Then data not represented by this model has to be regarded as corrupted by an additional non-zero mean noise process. It is shown in the Section 3 that model estimation for piecewise data can be achieved only by high breakdown point robust techniques which tolerate non-zero mean noise.

Least squares polynomial surface estimation of piecewise constant data (i.e., containing a step edge) has an important property of which we make use later. Let the step edge in the data be of height h and corrupted

with noise having standard deviation σ. We estimate the surface in a window centered on the discontinuity. The simplest, degree-0 model is that of a constant plane ($p = 1$). The least squares estimated value yields a plane lying between the two original horizontal surfaces that form the step edge. When a planar, degree-1 model ($p = 3$) is used, the estimated plane lies across the edge and intersects both surfaces.

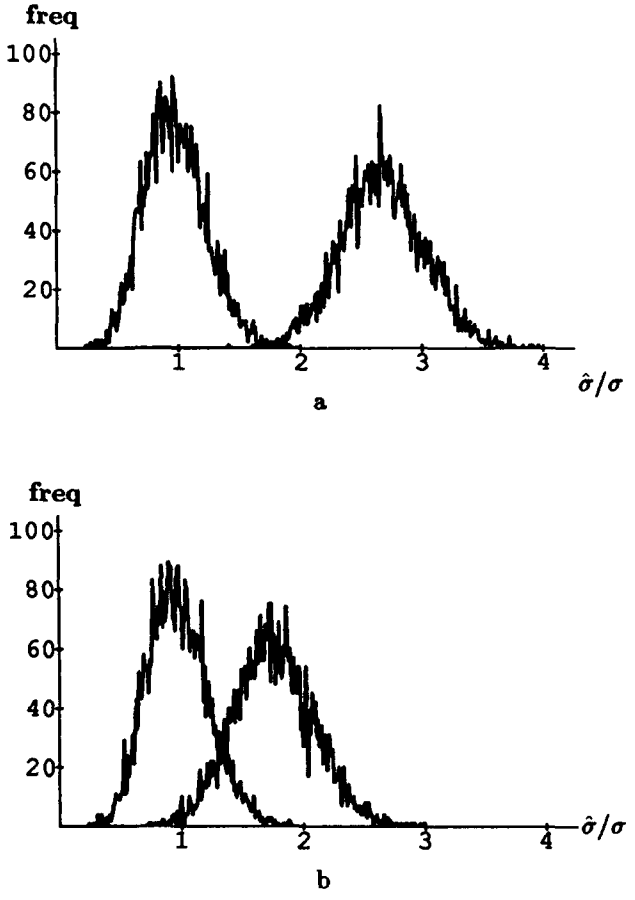


Figure 1: Comparison between the normalized standard deviation estimates computed with least squares (LS) estimators. Left histogram: constant surface. Right histogram: step edge. a) Degree-0 LS, SNR = 25. b) Degree-0 LS, SNR = 8.

Both fits are erroneous and do not correspond to either of the original surfaces. However, up to moderate signal-to-noise (SNR) ratios, when the standard deviation of the residuals (the noise's standard deviation estimate) is computed the degree-0 model yields an increase relative to σ while the degree-1 model may not. Thus, when a window operator performing degree-0 least squares estimation (averaging) slides along piecewise constant noisy data, the standard deviation estimate increases in the neighborhoods of step edges. The increase is significant enough to be used for coarse detection of the presence of a model discontinuity (edge).

To illustrate the effect, the standard deviation of the residuals was computed in a 3×3 window for constant and planar models. The data was either a two-dimensional step edge with amplitude h or a horizontal surface, both corrupted with i.i.d. Gaussian noise $(0, \sigma^2)$. The signal-to-noise ratio is defined as SNR$= (h/\sigma)^2$. To facilitate comparison the estimated standard deviation is normalized by σ.

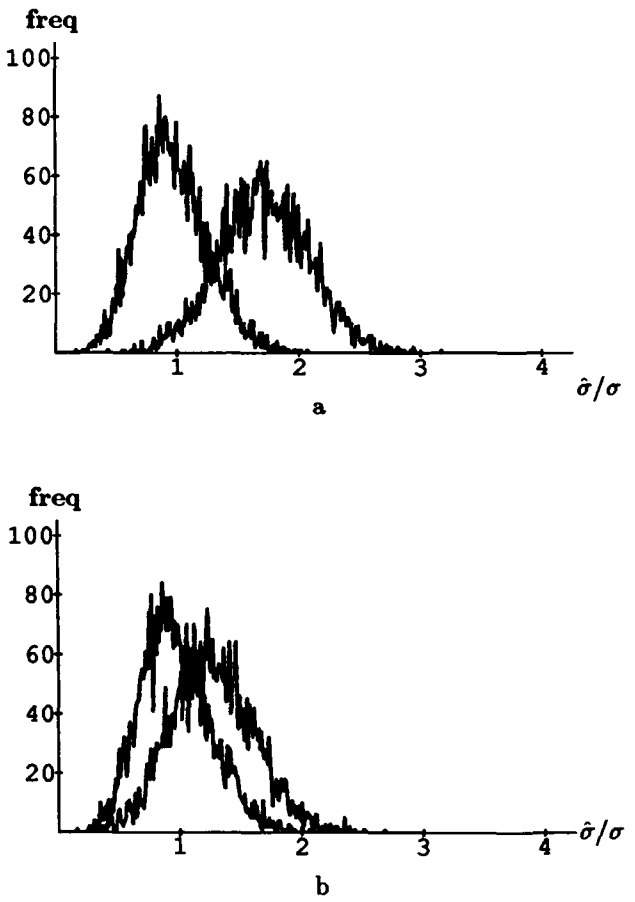


Figure 2: Comparison between the normalized standard deviation estimates computed with least squares (LS) estimators. Left histogram: constant surface. Right histogram: step edge. a) Degree-1 LS, SNR = 25. b) Degree-1 LS, SNR = 8.

In Figure 1 the results are presented as histograms of the normalized standard deviation estimates computed from 5000 trials for SNRs of 25 and 8. Each figure contains two overlaid histograms. The peak at the left always corresponds to the window containing the horizontal surface, and the peak at the right to the window containing the step edge. When the signal-to-noise ratio is high (SNR$= 25$) and the degree-0 model ($p = 0$) is used, the two histograms are clearly separated (Figure 1a). Thus a reliable indication of the presence of a discontinuity is obtained. This result is widely applied for adaptive filtering in the signal processing literature.

For an application in computer vision see [10].

If a degree-1 model ($p = 3$) is used a slight overlap of the two histograms appears (Figure 2a) which reduces the probability of discontinuity detection. A similar overlap appears with the degree-0 model only at SNR = 8 (Figure 1b). At such signal-to-noise ratios the degree-1 model already yields almost completely overlapping histograms (Figure 2b). Note that for constant surfaces the degree-0 least squares standard deviation estimates are unbiased, i.e., the peaks of their histograms are always close to one. (The degree-1 model yields underestimates since it absorbs some of the noise variations into the model.) The use of larger (say 5×5) windows decreases the spread of the degree-0 estimates but also shifts the histogram for the edge downward.

To conclude this section we emphasize two observations:

- When all the data belongs to one model corrupted with zero mean noise the least squares estimates are unbiased but may have a large standard deviation.
- At moderate signal-to-noise ratios the degree-0 least squares estimate of the noise's standard deviation can be used as warning about the presence of a step edge in the data.

These two properties will be of importance for the robust CBD estimator proposed in Section 4.

Assumptions: The model is defined by the parameters β_k, $k = 0 \ldots (p-1)$. The data contains n points with a fraction $\epsilon < 0.5$ of outliers.

1. Randomly select a p-tuple Π from the data.
2. For this p-tuple compute the parameter values $\hat{\beta}_k(\Pi)$ by solving the p equations. Note that if a linear model is assumed the p-tuple must yield a full rank coefficient matrix for the system.
3. Retain all parameters but $\hat{\beta}_0(\Pi)$. Project the data into the β_0 subspace by computing

$$\alpha_i(\Pi) \triangleq z_i - \sum_{k=1}^{p-1} \hat{\beta}_k(\Pi) x_k(i) \quad i = 1 \ldots n. \qquad (9)$$

4. Find the mode of the $\alpha_i(\Pi)$ sequence and allocate it to $\hat{\beta}_0(\Pi)$. Store the corresponding shortest half-window size

$$\delta(\Pi) = \left[\min_{\hat{\beta}(\Pi)} \operatorname{med}_i r_i^2 \right]^{1/2}. \qquad (10)$$

5. Repeat Steps 1 to 4 q times. The final LMedS estimates are from the p-tuple yielding $\delta = \min_{\Pi} \delta(\Pi)$.

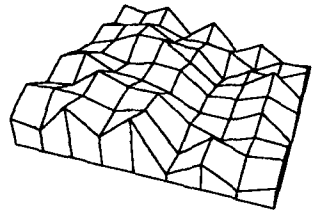
Figure 3: The LMedS Algorithm

3 The Least Median of Squares Estimator

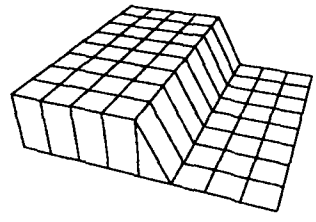
In this section we discuss a robust estimator which can handle weakly corrupted piecewise data. For convenience we describe the one-dimensional case; extension to multi-dimensions is immediate.

3.1 Mean, Median and Mode

The ability of an estimator to handle severely corrupted data is captured by its *breakdown point* ϵ^*. This is the smallest fraction of the data which can yield arbitrary estimate values. For example, the degree-0 least squares estimate (the *mean* of n data points, z_i) has $\epsilon^* = 1/n$, since one large valued erroneous point already compromises the result. The asymptotical breakdown point of the degree-0 LS estimator thus is 0. The zero breakdown point property is common for all least squares estimators [2, p. 328].



a



b

Figure 4: Piecewise constant data. a) Uncorrupted data. b) Noisy data, SNR = 8.

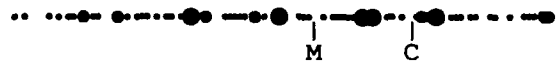


Figure 5: Ordered sequence of the pixel values on the noisy piecewise data

Two types of severely corrupted data can be distinguished. *Outliers* are data points with values $z_{i,j}$ which

piecewise data can always be described by one model and a set of outliers. For example, one of the planes forming a roof edge is taken as the model and the points belonging to the other are then regarded as outliers relative to that model. Once zero mean noise is also present, however, the dichotomy between the model and the outliers may not be easy to establish. (We return to this subject in Section 3.3.)

Leverage points are data points corresponding to outlying regressor variable values $x_k(i,j)$. These points are not necessarily "bad" since their values may be close to the value predicted by the model. However, if a data point is far away from the rest and cannot be accounted for by the model, it may exert an increased influence on the estimate. The sparse data in some computer vision applications (for example, stereo) may contain leverage points which compromise the result in spite of the fact that their value is within the range of the data. Detection of leverage points requires special techniques. See for example [14] for a method using an estimator similar to the one to be discussed in Section 3.2.

The breakdown point of an estimator cannot exceed 0.5 without making use of a priori information about the data. Indeed, at least half of the data should be represented by the model to be the unique solution of the estimation. The *median*, the degree-0 least absolute deviation estimator (belonging to the L_1 family of estimators), tolerates close to half the data being severely corrupted. At the limit, the median has $\epsilon^* = 0.5$. It can be shown, however, that higher degree surface fitting in L_1 is sensitive to leverage points and it has $\epsilon^* = 0$ [2, p. 328].

Another degree-0 estimator is the *mode* of the data. For a continuous probability distribution function (p.d.f.) the mode is the most probable variable value, i.e., the maximum of the p.d.f. (Without loss of generality we can consider a unimodal p.d.f. for the moment.) Assume that n outcomes were obtained from the p.d.f. and they are ordered by increasing values. We first compute the half-length of a window containing $N \leq n/2$ points in all the $n - N + 1$ positions along the ordered sequence. (The half-length is defined as half the difference between the data point values at locations i and $i + N - 1$ in the ordered sequence.) The midpoint of the shortest window is taken as the mode since the maximum of the p.d.f. implies the most outcomes in an interval.

Let the window size equal to half the data size, i.e., $N = \lfloor n/2 \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function. Then it can be shown that the mode $\hat{\beta}_0$ minimizes the median of the squared residuals, i.e., satisfies

$$\min_{\hat{\beta}_0} \operatorname*{med}_i (z_i - \hat{\beta}_0)^2 \tag{11}$$

and that the value of (11) is the squared half-length of the window yielding the mode [13, p. 166]. The use of squares instead of absolute values assures the uniqueness of the solution for n an even number [13, p. 170]. The presence of the median in (11) makes the mode a robust degree-0 estimator with breakdown point close to 0.5.

3.2 Properties of LMedS

The minimization problem (11) can be generalized to an arbitrary model $\hat{\beta}$ generating the residuals r_i

$$\min_{\hat{\beta}} \operatorname*{med}_i r_i^2. \tag{12}$$

The expression (12) defines the *least median of squares* (LMedS) estimator introduced in statistics by Rousseeuw in 1984. The already mentioned book of Rousseeuw and Leroy [13] gives an excellent practical analysis of the estimator. The finite sample breakdown point of the LMedS estimator is

$$\epsilon^* = \frac{\lfloor n/2 \rfloor - p + 2}{n} \tag{13}$$

and asymptotically yields 0.5.

Since (12) has no analytical solution, to find the LMedS estimates a numerical technique, projection pursuit, is employed.

The search procedure (Step 5) has the goal of finding at least one p-tuple containing only data points represented by the model, i.e., *inliers*. Projection of such a p-tuple into the β_0 space (Step 3) can yield at most ϵ outliers. The mode detection procedure (Step 4) then assures the recovery of $\hat{\beta}_0(\Pi)$, and that the entire parameter vector satisfies (12).

The random sampling (Step 1) reduces the complexity of the LMedS algorithm to a manageable amount of computation. Mode computation per p-tuple requires $O(n \log n)$ operations and there are $O(n^p)$ possible p-tuples. Let the tolerated probability of error in random sampling be $Q \ll 1$. Then the probability that for q independently selected p-tuples at least one contains only inliers is

$$1 - [1 - (1 - \epsilon)^p]^q \leq 1 - Q. \tag{14}$$

From (14) q can easily be computed. For example, when a degree-1 model ($p = 3$) is fit to piecewise data with $\epsilon = 0.45$, and the tolerated probability of error is $Q = 0.01$, choosing $q = 26$ different 3-tuples suffices to find the three LMedS estimates. Note that q does not depend on the data size. Since the LMedS estimates correspond to one of the p-tuples, the only characteristic of interest about the structure of the data is ϵ.

A "hidden" assumption behind the LMedS estimator is that the inliers are only weakly corrupted by i.i.d. Gaussian noise. That is, the parameter values obtained from a p-tuple containing only inliers are close to the correct ones. The LMedS algorithm also returns a robust estimate of the noise's standard deviation:

$$\hat{\sigma} = 1.4826 \left(1 + \frac{5}{n - p}\right) \delta \tag{15}$$

where the term $5/(n - p)$ is the finite sample size correction [13, p. 202] and 1.4826 is the correction factor for median based Gaussian noise standard deviation estimates. Once the noise corrupting the model is defined (zero mean normal with variance $\hat{\sigma}^2$) the data can be classified into inliers and outliers. The weights

$$w_i = 1 \quad \text{if } |r_i| \leq 2.5\hat{\sigma} \tag{16}$$

mark the inliers with about a 0.95 confidence level (the uncertainty is due to the finite data size). If the nature of the noise is known a priori other correction factors and thresholds can be computed. The LMedS algorithm and the inlier/outlier dichotomy procedure, however, become unreliable when significant noise corrupts the data. This problem is the subject of the next section.

An optimization principle similar to the one used in the LMedS algorithm was already proposed for computer vision applications in [1] as the Random Sample Consensus (RANSAC) paradigm. In RANSAC the model derived from a p-tuple determines the set of data points agreeing with it within a given tolerance limit. If the number of points in the set exceeds a threshold, no more p-tuples are drawn. Thus RANSAC requires the a priori information of a tolerance limit and a cardinality threshold. LMedS automatically selects the model representing at least half the data by minimizing a quality measure of the fit, the median of the squared residuals. For a more detailed comparison of the two techniques see [5].

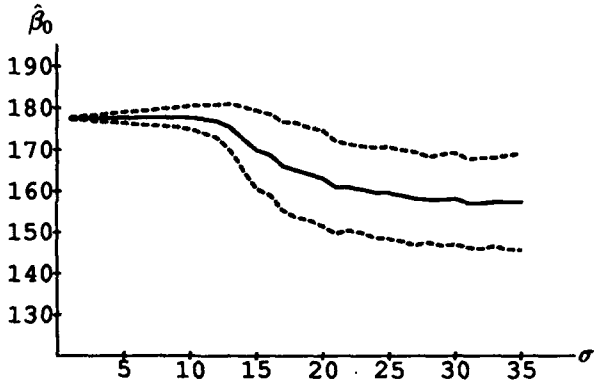


Figure 6: LMedS: $\hat{\beta}_0$ vs. σ.

3.3 LMedS in Computer Vision

The high breakdown point of the LMedS estimator makes it an ideal operator for handling piecewise data. The estimator identifies the model corresponding to the majority of the points in the set and discriminates the points not belonging to it, i.e., the outliers. The LMedS algorithm (as presented above) was applied to several computer vision problems. Kim et al. [3], Roth and Levine [12], Sinha and Schunck [15] have used it in surface reconstruction; Kumar and Hanson [4] in camera position estimation; Tirumalai et al. [17] in dynamic stereo.

We have already emphasized the necessary condition for obtaining "good" LMedS estimates, the existence of at least one p-tuple from which reliable model parameters can be extracted. When significant zero mean noise corrupts *all* the data points such a p-tuple may not exist. In this case a p-tuple containing only noisy inliers yields, like the least squares estimator (see Section 2.2), unbiased estimates with large variance. The noise can also

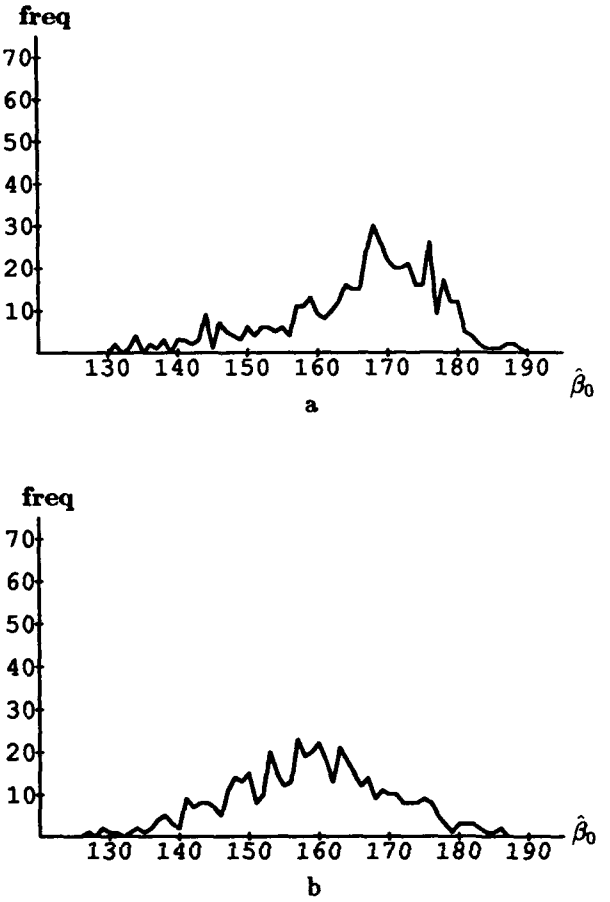


Figure 7: Histograms of LMedS estimators. a) $\sigma = 17$ (SNR = 8). b) $\sigma = 25$ (SNR = 4).

erase the clear dichotomy between data points belonging to the model and outliers.

Consider again piecewise constant data containing a step edge of amplitude h (ϵ close to 0.5) corrupted with zero mean noise taking values in the range $(-a, a)$. This definition is for convenience only; for Gaussian noise a could be, say, 3σ. Suppose also for the moment that $a \leq h/2$.

First we use the degree-0 LMedS estimator, i.e., compute the mode of the data. Assume that we have succeeded in recovering the uncorrupted value of the horizontal surface corresponding to the majority of the pixels. Then the squared residuals are distributed as follows: slightly more then half in the interval $(0, a^2)$ and the rest in the interval $[(h-a)^2, (h+a)^2]$. The estimate of $\hat{\sigma}$, the model noise standard deviation, is proportional to the square root of the median of squared residuals, (10) and (15), and thus it will exceed a. But a is the range of the noise and therefore $\hat{\sigma}$ is a strong overestimate. Large $\hat{\sigma}$ reduces the weights (16) and outliers may be classified as inliers.

When a degree-1 LMedS estimator is applied to the noisy piecewise constant data the large variance of the

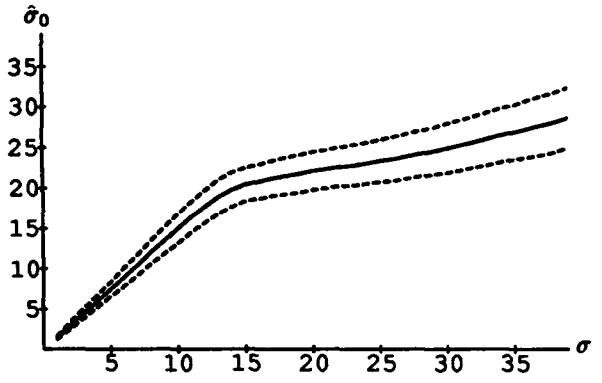


Figure 8: LMedS: $\hat{\sigma}$ vs. σ.

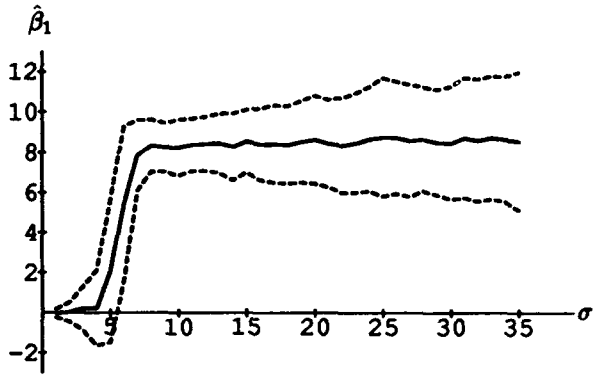


Figure 9: LMedS: $\hat{\beta}_1$ vs. σ.

3-tuple based estimates makes any plane orientation possible. Assume that we have chosen two 3-tuples, one generating the correct horizontal fit, the other yielding a plane tilted at about 45 degrees across the edge. The latter fit is similar to what a least squares estimator would have recovered from the window. If significant noise is present (a is close to $h/2$), most of the squared residuals for the tilted plane fit are between 0 and a^2. The median of this residual sequence is significantly lower than the median of the sequence obtained when the horizontal fit is employed (see above). The LMedS criterion (12) seeks the minimum and therefore the tilted plane is preferred. Note that tilted planes are always obtained from some 3-tuples since any sample containing pixels from both surfaces can yield one. The estimator will choose the correct, horizontal plane only when the amplitude of the corrupting noise is very small relative to the amplitude of the step, that is, the SNR is very high. The correct fit, however, implies an overestimate of σ as was shown above.

For noisy data with a discontinuity the mode detection procedure can also fail. Such data yields a bimodal distribution. Figure 4b shows noisy piecewise constant data having SNR = 8 derived from the ideal data in Figure 4a. The ordered sequence of pixel values is given in Figure 5. Since the resolution of the drawing does not allow distinct representation, for every data point larger circles stand for more points with similar values.

The value of the constant surface corresponding to the majority of the pixels in the uncorrupted data (Figure 4a) is marked 'C' in the sequence. The significant noise level spreads the values across the entire range of the data and no clear bimodality can be seen. The mode detection procedure finds that the shortest window corresponds to a value between the two surfaces forming the original step edge. The location of the mode detected for the noisy data is marked 'M'. The estimate obtained is very similar to what a degree-0 least squares estimator would recover.

To illustrate the severity of the artifacts of the LMedS estimator for piecewise constant data we have performed a simulation study. A step edge of amplitude $h = 50$ was defined on a dense square grid in a 9×9 region. Four columns had the value 128 and five had 178. Gaussian noise (i.i.d.) with standard deviation from $\sigma = 0$ to $\sigma = 35$ (SNR = 2) was added to the data. For each σ value 100 trials were run. The original data thus has $\epsilon = 0.44$, and a high breakdown point estimator should recover from the noisy piecewise constant data an estimate close to 178.

First the behavior of the degree-0 LMedS estimator, i.e., mode detection, is analyzed. In Figure 6 the dependence of $\hat{\beta}_0$, the estimated mode value, on σ, the noise's standard deviation, is shown. The average value of 100 trials as a function of σ is drawn with a continuous line. As a measure for the "stability" of the estimate *its* standard deviation for the trials was also computed. The dashed lines in Figure 6 show one standard deviation distance from the average function of σ.

The estimated mode value is satisfactory for high signal-to-noise ratios but deteriorates for $\sigma \geq 14$ (SNR$\leq$ 13). Higher noise more often yields erroneous mode detection and the average decreases. Two histograms of $\hat{\beta}_0$ for a given σ based on 500 trials are shown in Figure 7a ($\sigma = 17$, SNR = 8) and Figure 7b ($\sigma = 25$, SNR = 4). The estimates have a wide distribution and the complete failure of the degree-0 LMedS estimator at such noise levels is clearly seen.

The effect of overestimation of $\hat{\sigma}$, the noise's standard deviation, by the degree-0 LMedS estimator is shown in Figure 8 where the dependence of $\hat{\sigma}$ on σ is given using the same conventions as in Figure 6. The linear increase for high SNR is already an overestimation since it has slope 1.5 instead of 1. At significant noise levels the failure of the degree-0 LMedS estimator to provide the correct model yields incorrect $\hat{\sigma}$ estimates.

We have shown above that for noisy piecewise constant data the degree-1 LMedS estimator (planar fit) will favor a tilted plane over the correct horizontal surface. In Figure 9 the dependence of the slope estimate $\hat{\beta}_1$ on σ is shown in the same experimental conditions as before. In the presence of a very low noise level ($\sigma = 4$, SNR = 150) the estimate already deviates from the correct value of $\beta_1 = 0$. The deterioration of the performance is very fast; at $\sigma = 7$ (SNR = 50) $\hat{\beta}_1$ already reaches the plateau corresponding to a plane tilted across the edge.

We have shown in this section that application of

LMedS to noisy piecewise constant data with a large fraction of outliers can result in failure.

- The high breakdown point property of the LMedS estimator no longer is satisfied and performance similar to least squares is often obtained.
- The LMedS estimator cannot recover the correct model when this is misspecified, i.e., non-essential parameters are introduced.
- The inlier/outlier dichotomy can become erroneous because of overestimation of the noise's standard deviation.

In the next section we propose a new robust estimation technique which extends the high breakdown point property to piecewise data with low signal-to-noise ratios. It also achieves a significant speedup relative to LMedS while reducing the above mentioned deficiencies.



Figure 10: The censored mean distribution of the pixel values on the noisy piecewise data. The mode 'M' is detected with the enhanced procedure.

4 The Consensus by Decomposition Estimator

The principle behind the LMedS estimator (and the RANSAC paradigm) can be described as follows:

> Compute a candidate model based on a randomly chosen small subset of the data. Apply this model to all the data. Compute a global quality measure for the model. Optimize the quality measure by repeating the procedure several times.

To have reliable LMedS estimates it is necessary that at least one candidate model carry the correct parameter values. We have shown that when this condition is not satisfied the LMedS estimator loses most of its attractive properties. The high breakdown point of an estimator can be interpreted as seeking the consensus in the data about the assumed model. Consensus in LMedS is achieved by minimizing the criterion (12) and in RANSAC by maximizing the number of inliers.

However, consensus can be achieved in other ways. In this section we describe a new, high breakdown point robust technique, which we call the *consensus by decomposition* (CBD) estimator. The concept of *sampling structure*, defined as the set required to index the data, is of importance for the CBD estimator. For example, the sampling structure of two-dimensional spatial data described in Cartesian coordinates is the set of (x, y) addresses of the points at which data is available. The principle behind the CBD estimator differs from that of LMedS:

> Compute a candidate model for a small neighborhood of every data point in the sampling structure. Define a separate estimated image for each of the model parameters. Estimate the parameters from their images by recursively applying an enhanced mode detection procedure.

The necessary condition for a reliable CBD estimate is that the majority of the candidate models carry unbiased parameter estimates. This condition is much weaker than the one required by LMedS and makes possible the elimination of the latter's artifacts. The CBD estimator was described briefly in [8].

Assumptions: The data is a step edge (ϵ close to 0.5) homogeneously corrupted by zero mean noise with standard deviation σ. The size of the neighborhood used for local processing allows an absolute majority of unbiased estimates.

1. (Smoothing.) Using a degree-0 LS estimator, compute the local mean and standard deviation estimates for small neighborhoods around every data point in the sampling structure.
2. Using a degree-0 LMedS estimator, estimate the noise's standard deviation $\hat{\sigma}$ as the mode of the standard deviation distribution. Mark the outliers of this distribution.
3. (Censoring.) Eliminate the points found as standard deviation outliers from the mean estimate distribution.
4. Using a degree-0 LMedS estimator, estimate the value of the surface model $\hat{\beta}_0$ as the mode of the censored mean distribution.
5. Define the inlier/outlier dichotomy of the censored mean distribution. Map it into the data. Eliminate possible inconsistencies on the sampling structure. Extend the dichotomy to the entire set of data.

Figure 11: The EMD Algorithm

4.1 Mode Detection with Smoothing and Censoring

The failure of mode detection (the degree-0 LMedS estimator) for noisy piecewise constant data and the problem of overestimating the noise's standard deviation were discussed in Section 3.3. Since mode detection is the computational module assuring the robustness of LMedS, we must first improve its performance if want to robustly estimate noisy piecewise data.

Consider piecewise constant data containing a step edge (ϵ close to 0.5) corrupted with zero mean noise having standard deviation σ. We assume a homogeneous

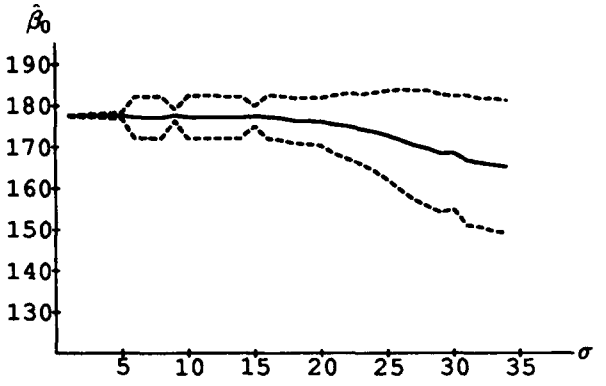


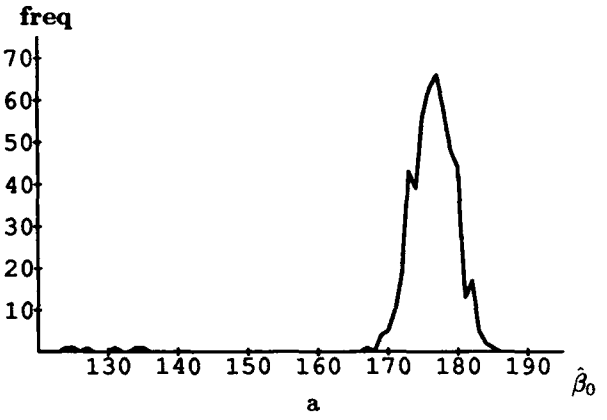
Figure 12: EMD: $\hat{\beta}_0$ vs. σ.

noise process, that is, the same process corrupts both sides of the edge. This assumption is a weak one and is practically always satisfied in computer vision applications. A small neighborhood is defined on the sampling structure around every data point. (Border effects can be eliminated by using only a subset of the available data.) In each of these neighborhoods a degree-0 least squares (LS) estimator is applied and the estimates for the local mean and standard deviation of the data (and also the noise!) are computed.

In Section 2 we have shown that for neighborhoods not containing the edge both LS estimates are unbiased. In neighborhoods that cross the edge (thus having data from both surfaces) the mean estimate has an intermediate value and the standard deviation estimate increases. To obtain the distribution of each of the two LS estimates their values are ordered into two sequences. We want the unbiased estimates to become the absolute majority in these sequences. Since the two LS estimates are defined on the same sampling structure as the data, we achieve our goal if the size of the data is large enough relative to the size of the neighborhood.

The distribution of the standard deviation estimates is unimodal with the mode close to the correct value σ. Estimates from neighborhoods containing the edge are placed at the upper tail of the ordered sequence. The mode detection procedure described in Section 3.1 can always cope with such a distribution and thus the mode is a reliable estimate of the corrupting noise's standard deviation. The mode detection procedure also returns a robust measure for the spread of the distribution (15). By using the weights (16) we can mark the estimates which are far away from the correct standard deviation value, i.e., the outliers. Most of the outliers from the upper tail correspond to the data points with neighborhoods containing the edge.

Like the ordered sequence of the original data, the distribution of the local mean estimates is bimodal. The two modes correspond to the values of the two original surfaces. Note that we must have the majority of LS estimates in the mean sequence unbiased but not all of these estimates have to represent the same surface. The replacement of the data by a local mean reduces the spread



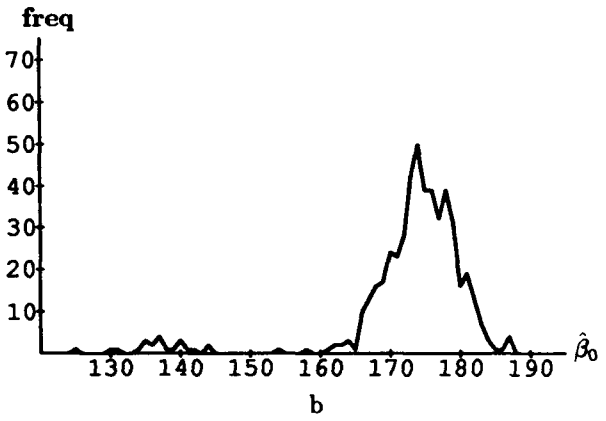
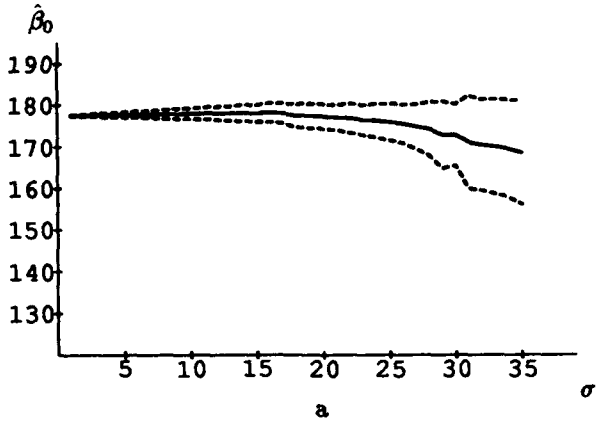


Figure 13: Histograms of EMD estimates a) $\sigma = 17$. b) $\sigma = 25$.

of the estimates around the two modes. The data points where the averaging yields incorrect (biased) results are near the edge and were already marked as outliers in the standard deviation sequence. These points thus can be removed and the two modes of the remaining (censored) mean sequence become well separated. The mode detection is now reliable and the mode corresponding to the majority of the data points is recovered. Compare the censored mean sequence in Figure 10 with the sequence corresponding to the original data in Figure 5. The new, enhanced mode estimate (marked 'M') is close to the correct surface value (marked 'C').

The enhanced mode detection (EMD) procedure for noisy piecewise constant data is given in Figure 11.

The experiments described in Section 3.3 analyzing the behavior of the degree-0 LMedS estimator were repeated for the EMD procedure. The local neighborhood was of size 3×3. In Figure 12 the dependence of $\hat{\beta}_0$, the estimated mode value, on σ, the noise's standard deviation, is shown. The performance of the EMD procedure no longer deteriorates for $\sigma \geq 14$ (SNR$\leq$ 13) as that of LMedS did (Figure 6). The standard deviation of $\hat{\beta}_0$ for the 100 trials, however, appears to be larger than in the



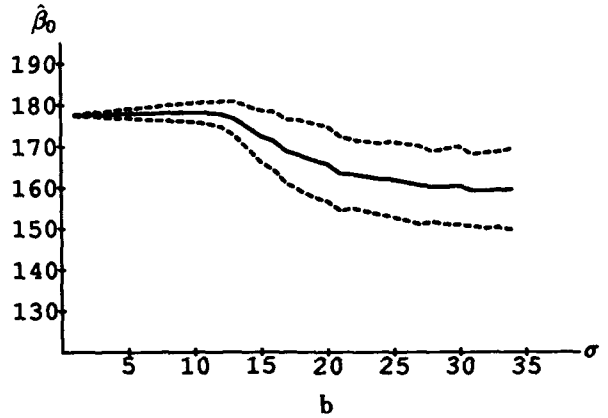


Figure 14: Graphs of $\hat{\beta}_0$ vs. σ with the lowest 12 percent of the values trimmed. a) EMD. b) LMedS.

case of LMedS.

The phenomenon is explained using the histograms of 500 $\hat{\beta}_0$ estimates for a given σ. They are shown in Figure 13a (σ = 17, SNR= 8) and Figure 13b (σ = 25, SNR= 4). In the EMD procedure most of the trials yield a $\hat{\beta}_0$ estimate close to the correct value β_0 = 178 (compare with Figure 7). The estimates no longer spread over the entire range of the data, but in a few experiments mode values in the vicinity of the other surface (β_0 = 128) are detected. These artifacts are caused by the probabilistic nature of the censoring process. The noisy data points derived from the two surfaces are almost equally represented (ϵ = 0.44). After the censoring process the majority in the mean sequence may "flip" toward the second surface and the mode detection returns the corresponding value. Note in Figure 13a the distribution of "flipped" estimates close to the other surface's value. The outlying mode values dramatically increase the standard deviation of $\hat{\beta}_0$ for the trials. In Figure 12 only two σ values yielded 100 trials without artifacts where small standard deviations were obtained (σ = 9 and σ = 15).

In most computer vision applications the existence of

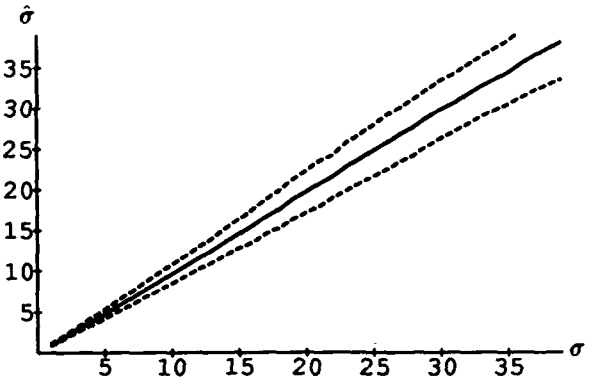


Figure 15: EMD: $\hat{\sigma}$ vs. σ.

such a "flip" is of no importance since the inlier/outlier dichotomy of the data obtained at Step 5 of the EMD procedure is correct albeit reversed. Cooperative processes among adjacent (or overlapping) data sets will thus always converge to the same final result. To remove the effects of the artifacts we have trimmed the lowest 12 percent of $\hat{\beta}_0$ values from the trials. The dependence of the trimmed $\hat{\beta}_0$ on σ (Figure 14a) has practically the same mean as before but with a reduced standard deviation. Note that the probability of a "flip" is very low since discarding the lowest 12 percent eliminated most of the "flips" over the entire range of σ. Applying the same trimming method to the LMedS trials has no significant effect on the curve. Compare Figures 6 and 14b.

At Step 2 of the enhanced mode detection procedure the estimate $\hat{\sigma}$ of the noise's standard deviation is also obtained. Its value as a function of σ is shown in Figure 15. The dependence is now linear over the whole range and has slope one, i.e., the estimate is always correct. Compare with Figure 8.

We conclude from these experiments that the enhanced mode detection procedure succeeds in preserving the high breakdown point properties for piecewise constant data up to low signal-to-noise ratios. The procedure also returns a correct estimate for the noise's standard deviation. We can now proceed to develop a technique for the general case of piecewise data.

4.2 Robust Estimation of Noisy Piecewise Data

The performance improvement of the enhanced mode detection method relative to the original procedure is made possible through computing mostly unbiased estimates and robustly analyzing their distributions. For piecewise constant data the neighborhoods not containing a discontinuity (the step edge) yield the unbiased estimates. Unfortunately, for the general piecewise model this is not true for all the parameters. The local neighborhood computations can be regarded as translating a small window across the data. We are interested in the set of parameters invariant under translation over data coming from the same model. Only these parameters yield unbiased estimates and therefore only their dis-

tribution pooled together from several neighborhoods is meaningful. The discussion for arbitrary regressor variables $x_k(i,j)$ and/or sparse sampling structures is beyond the scope of this paper. In what follows we restrict ourselves to the case

$$x_k(i,j) = i^{k_1} j^{k_2} \quad k_1, k_2 = 0, 1, \ldots \qquad (17)$$

Assumptions: The piecewise data ($0 \leq \epsilon < 0.5$) is homogeneously corrupted by zero mean noise with standard deviation σ. The model has the parameters β_k, $k = 0, 1, \ldots, (p-1)$. The size of the neighborhood used for local processing allows an absolute majority of unbiased estimates for the translation invariant parameters.

1. (Spatial decomposition.) Using an LS estimator, compute a candidate model for a small neighborhood around every data point in the sampling structure.
2. (Parameter decomposition.) For each β_k, define a separate parameter image on the sampling structure.
3. (Consensus seeking.) Apply the enhanced mode detection procedure to the translation invariant parameter images. The resulting mode values $\hat{\beta}_k$ are the final CBD estimates for this subset of parameters.
4. Mark on the sampling structure the points that are inliers in all the translation invariant parameter images processed in Step 3.
5. Project the data into the subpace of the remaining parameters by using a relation similar to (9). Define the corresponding reduced model.
6. Repeat Steps 1 to 5 for the marked points until no parameter is left unestimated.

Figure 16: The CBD Algorithm

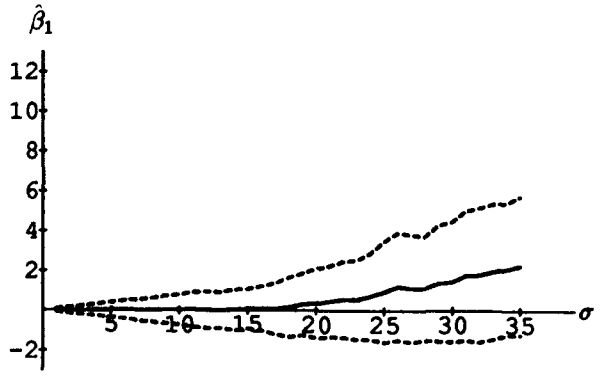


Figure 17: CBD: $\hat{\beta}_1$ vs. σ.

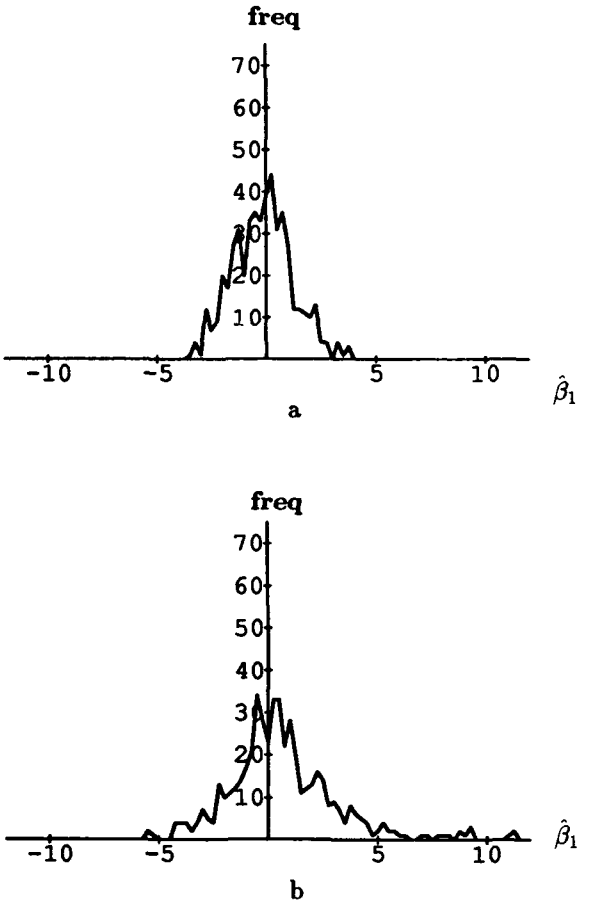


Figure 18: Histograms of CBD estimates of $\hat{\beta}_1$ a) $\sigma = 17$. b) $\sigma = 25$.

Model (1) is then that of a polynomial surface and we perform robust surface fitting on a dense Cartesian grid.

The linear model (1) used by local unweighted least squares estimation is correct when no discontinuity is present. Since the estimation is unweighted the covariance matrix of the estimates (6) becomes $\mathrm{cov}\left[\hat{\beta}\right] = \sigma^2(\mathbf{X}^t\mathbf{X})^{-1}$. The matrix $\mathbf{X}$ is defined by the values of the regressor variables. We want $\mathbf{X}$ to be an orthogonal matrix, that is, to satisfy the property $\mathbf{X}^t = \mathbf{X}^{-1}$. In this case the covariance matrix (6) is diagonal and the estimates are uncorrelated.

In our case orthogonality of $\mathbf{X}$ is achieved if the model is defined in terms of a polynomial base instead of the regressor variables (17). That is, we use linear combinations of the monomials (17). These basis polynomials are known as Chebyshev polynomials and their properties in the context of computer vision are discussed in [7]. The Cartesian product of the Chebyshev polynomials serves as the base for the two-dimensional model. The coordinate variables i or j belong to the Chebyshev base polynomials. This explains why the planar model (7) yields the diagonal covariance matrix (8). For a quadratic surface the second degree terms must be introduced by the

corresponding Chebyshev polynomials and not by i^2 or j^2 alone. In what follows we assume that the least square estimated parameters are uncorrelated.

Since the estimates are uncorrelated the estimation process can be decomposed into k independent procedures each having the partial model $\hat{z}_k(i,j) = \hat{\beta}_k x_k(i,j)$ [9, p. 137]. The relation between the regressor variables defined in two different neighborhoods can be found by performing the translation transformation $i \leftarrow (i - i_0)$ and $j \leftarrow (j - j_0)$. A parameter estimate $\hat{\beta}_k$ is invariant to translation if its partial model remains unchanged after the transformation. It is immediate to observe that only the parameters β_k multiplying the highest degree terms remain unchanged in the translated model. The rest of the polynomial coefficients in the translated model become linear combinations of the original β_ks. For example, when a planar (degree-1) model is used only the two slope parameters are invariants under translation; the intercept is not.

The uncorrelatedness of the estimates allows us to define for each parameter a separate *parameter image* on the sampling structure. Thus after performing the local LS estimations at every data point, p parameter images are defined. The fundamental observation behind the consensus by decomposition (CBD) estimator is that the translation invariant parameters of piecewise data yield parameter images which are blurred noisy step edges. Indeed, for data belonging to the same model the translation invariant parameters are unbiased estimates, i.e., constants corrupted with noise. Neighborhoods containing data from both models yield incorrect estimates, introducing blur at the step discontinuity. Note that the noise is correlated since the estimation is performed in overlapping neighborhoods. Thus the piecewise constant data case discussed in the previous section is an idealization of a translation invariant parameter image.

The consensus by decomposition algorithm (CBD) is given in Figure 16.

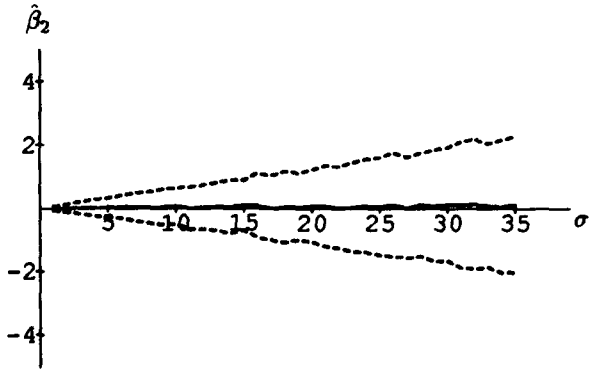


Figure 19: CBD: $\hat{\beta}_2$ vs. σ.

The CBD algorithm seeks a consensus on each model parameter by separately analyzing its underlying distribution with the enhanced mode seeking procedure. First the algorithm estimates the coefficients of the highest degree terms of the polynomial surface which are the translation invariant parameters. For a quadratic surface, for example, the coefficients are those of i^2, ij and j^2. These estimates are then used to compute the data values projected into the subspace of the remaining parameters. The projected data is represented by a reduced model. In the case of the quadratic example, the reduced model is planar. The reduced model uses the same regressor variables. The coefficients of the currently highest degree terms are the new set of translation invariant parameters. In our example they are the two slope components of the plane. Note that the projection propagates the estimation errors into the remaining parameters.

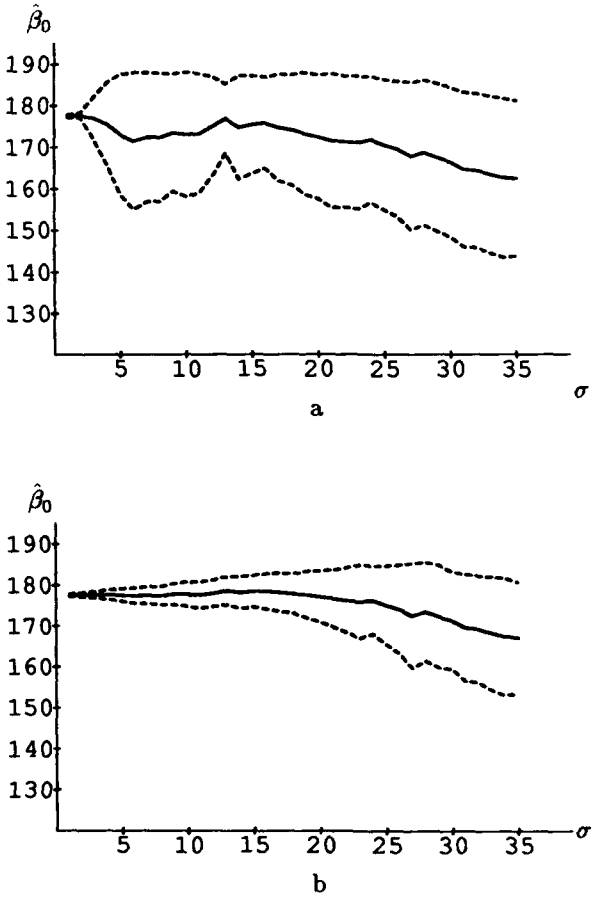


Figure 20: CBD: $\hat{\beta}_0$ vs. σ. a) All the values b) The lowest 12 percent trimmed

Only the inliers, i.e., data points belonging to the model, are taken into account in the projected data. It is possible that at Step 4 the intersection of the set of inliers in the different translation invariant parameter images yields only a few points. This can happen when the inlier/outlier dichotomy is "flipped" in some of the parameter images. As we have seen in Section 4.1 the probability of a "flip" is low. Cooperative processes among the parameter images help to eliminate the artifacts. After projection the local LS estimate use the reduced model and the consensus for the new transla-

tion invariant parameter set is sought. The entire cycle is then repeated until $\hat{\beta}_0$ is finally estimated.

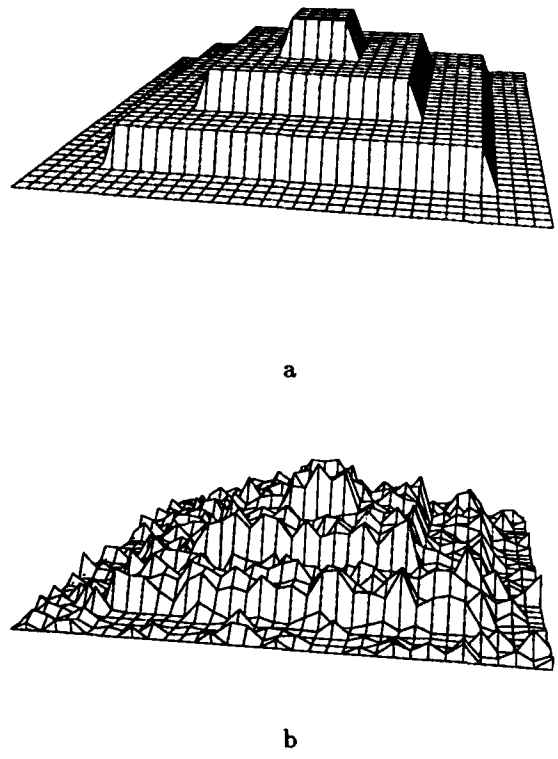
The behavior of the degree-1 CBD algorithm for piecewise constant data is analyzed next. The same experimental conditions were used as for the analysis of degree-0 estimators. The dependence of the slope across the edge, $\hat{\beta}_1$, on σ, the standard deviation of the corrupting noise is shown in Figure 17. Compare it with the behavior of LMedS shown in Figure 9. The CBD algorithm correctly returns a value close to $\beta_1 = 0$ up to low signal-to-noise ratios. The histograms of $\hat{\beta}_1$ shown in Figure 18a ($\sigma = 17$, SNR$= 8$) and Figure 18b ($\sigma = 25$, SNR$= 4$) illustrate the spread of the estimates. Note that no "flips" occur since for piecewise constant data the slopes have a unimodal distribution. The dependence of $\hat{\beta}_2$ on σ is shown in Figure 19. As expected the estimates are always close to the correct value $\beta_2 = 0$. The dependence of $\hat{\beta}_0$ on σ is shown in Figure 20a. The correct value is $\beta_0 = 178$. The $\hat{\beta}_0$ parameter image is piecewise constant and therefore the "flips" discussed in Section 4.1 may appear. When the lowest 12 percent of the $\hat{\beta}_0$ values in the 100 trials are trimmed, however, the dependence is as expected (Figure 20b). The error propagation discussed above slightly increases the standard deviation relative to the use of the degree-0 estimator (compare with Figure 14a).

It is important to notice that we have eliminated the problem of non-essential model parameters discussed in Section 3.3. In spite of using a degree-1 model for piecewise constant data we were able to recover the horizontal surface corresponding to the majority of the pixels. Given the data and a family of models (e.g., polynomial surfaces), the model order problem deals with the selection of the model best fitting the data. There is no unique solution for the problem and it is an active research area both in the context of classic [9, chapter 7] and robust [2, pp. 366–7] statistical techniques. Without claiming a theoretical advance, the CBD estimator may constitute a practical solution at least for the polynomial surface fitting case discussed here.

Another important observation is that of the speed-up achieved by the CBD estimator relative to the LMedS estimator employing random sampling. When q p-tuples are chosen randomly from n data points, the computational complexity of LMedS is $O(qn \log n)$. Due to the decomposition used by the CBD estimator the computational complexity decreases to $O(pn \log n)$. Since q is not a linear function of p (14) the speed-up is by an order of magnitude for planar surfaces ($p = 3$) and can reach two orders of magnitude for quadratic surfaces ($p = 6$).

The CBD estimator is superior to LMedS for most computer vision applications since it: 1) can handle noisy piecewise data; 2) can tolerate misspecified models; 3) is faster.

The only type of data for which our implementation of the CBD estimator may fail is data corrupted with impulse noise. When ϵ is close to 0.5 and the corrupted data points are uniformly distributed on the sampling structure, most of the local LS estimates become erroneous and unbiased estimate majorities cannot be obtained. The impulse noise, however, can be removed by preprocessing. The preprocessing can be either local LMedS interpolation, if no significant zero mean noise is present, or M-estimator based local smoothing (without guaranteed good performance!) if such noise is present.



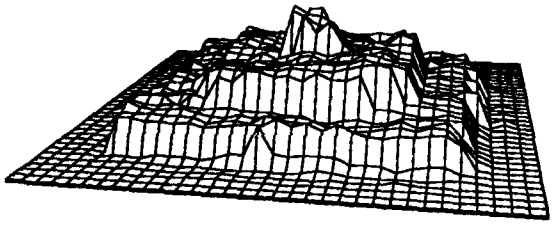
a b

Figure 21: a) A "wedding cake" b) A "wedding cake" with Gaussian noise (SNR = 11).

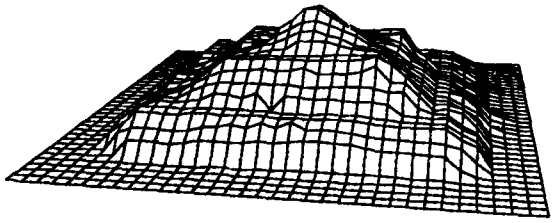
5 Applications

In this section we briefly present two applications that use the CBD estimator: image smoothing and edge detection. The use of robust techniques in low-level vision task is of importance since an improved output lessens the feature definition uncertainty in high-level vision. When appropriate we compare the results with those obtained by substituting LMedS for the CBD module.

The CBD algorithm is applied here to the case of robust polynomial surface fitting on a dense sampling structure. To discuss some of the implementational problems and to illustrate the performance of the technique on real data, the CBD algorithm was used as a computational module in two low-level vision tasks. The *image smoothing* application shows the superiority of CBD to LMedS when the data is corrupted by zero mean noise. The *simultaneous step/roof edge detection* application makes use of the capacity of the CBD algorithm to correctly estimate piecewise constant data with degree-1 estimators.



a



b

Figure 22: a) CBD based smoothing. b) LMedS based smoothing.

5.1 Image Smoothing

In centered window based image smoothing methods the value of a pixel is updated with the value of the intercept $\hat{\beta}_0$ since the pixel has local coordinates $(0, 0)$. High breakdown point estimators return the model corresponding to the majority of the pixels and this can yield artifacts. At corners the window center does not necessarily belong to the surface representing the majority of the pixels. Similarly, if the inlier/outlier dichotomy is reversed (for example, a "flip" occurs in the CBD estimation) the smoothed pixel gets an incorrect value. The latter case can appear only at low signal-to-noise ratios.

Figure 21a is wire-frame display of a synthetic "wedding cake" and Figure 21b is a display of the "wedding cake" with additive Gaussian noise (SNR = 11). The wire-frame representation was selected to better display the effects of the smoothing. Although the noisy data seems very noisy the amount of noise that was added to the original data seems harmless enough in a gray level representation.

Figures 22a and 22b show the results of running a centered window based smoothing operator on the noisy data using CBD and LMedS (respectively). In both cases a 9×9 window was used and 3×3 windows were used as local neighborhoods for CBD. Note how the CBD operator better preserves the edges while LMedS smooths

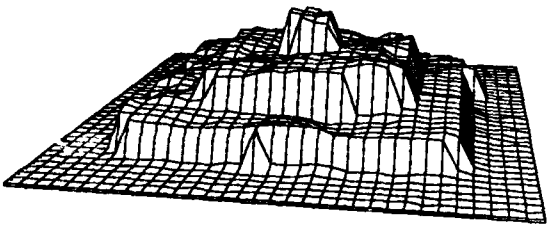


Figure 23: Results of two level consensus based smoothing.

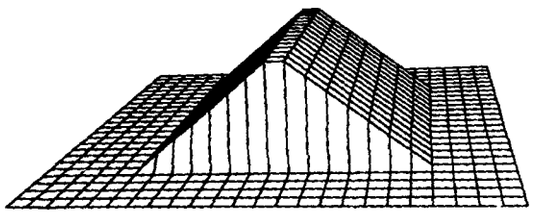
them. Some artifacts appear as discussed above.

These artifacts are eliminated if consistency among adjacent windows processing overlapping data is assured. Most of the windows recover the correct local structure and thus a consensus based decision allocates the correct smoothed value to the pixel. The result of such a procedure can be seen in Figure 23. Note that this consensus is hierarchically at a higher level that the one detected in the CBD estimator. Cooperation among windows is also helpful to keep the size of the windows using the CBD algorithm relatively small. Diagonally oriented discontinuities require larger windows in order to have the majority of local estimates unbiased. However, this worst case situation appears only when the discontinuity passes through the center of the window. Thus the windows containing a discontinuity with an offset still recover the correct local structure.

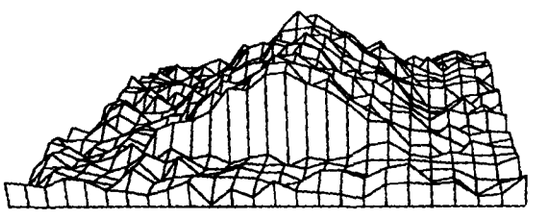
The same experiment was run on a synthetic roof (Figure 24a) with added noise having $\sigma = 4$ (Figure 24b). The results are shown in Figure 25. A consensus at a higher level is not needed since a "flip" occurring in a window centered on the rooftop does not change the smoothed value as it does at a step discontinuity, and a "flip" is less likely to occur in other windows.

The results clearly show the superiority of the CBD estimator over LMedS. It is highly probable that due to the noise, the LMedS search procedure will result in an LMedS value that corresponds to a model with different parameters from the original. Thus, the roof top is smoothed by the LMedS procedure. The CBD estimator considers only two alternatives for parameters per window and thus obtains the correct parameters.

As for running times, the CBD based smoothing procedure ran for 45 seconds on a DECstation while the LMedS procedure took over 11 minutes on the same machine. The CBD estimator thus achieves a speedup of 15 over the LMedS estimator. This agrees with the computational complexity comparison since $p = 3$ and 50 samples were taken in the LMedS procedure.

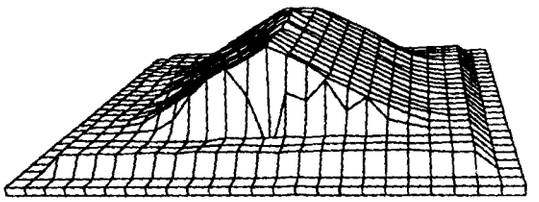


a

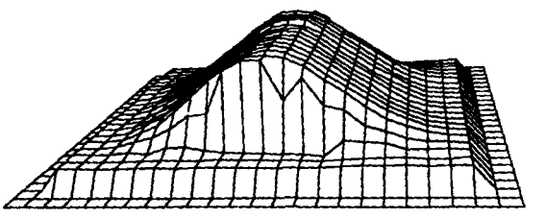


b

Figure 24: a) A synthetic roof. b) A synthetic roof with additive noise ($\sigma = 4$).



a



b

Figure 25: a) CBD based smoothing b) LMedS based smoothing.

5.2 Simultaneous Step/Roof Edge Detection

When a robust estimator is applied to uncorrupted piecewise polynomial data the boundary between the inlier and outlier regions is the location of a discontinuity. The delineated boundary location is invariant to a shift of the processing window as long as the discontinuity is within the window. In real image data the delineation of the boundary is not perfect due to noise and/or model inadequacy. Accumulation of the boundary candidate information obtained from a set of overlapping windows, however, can be used to allocate to every pixel in the image a probability of the presence of a discontinuity at that pixel. A pixel that was marked as an edge candidate in most of the overlapping windows is more likely to be an edge pixel than a pixel that only a minority of the overlapping windows marked as an edge candidate. Note that we again employ a second level of consensus to reduce the errors in the output.

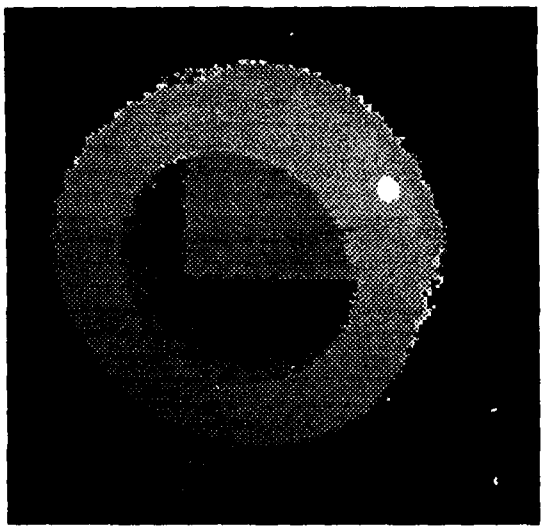
The higher level consensus is powerful enough to handle most of the errors introduced by LMedS and produce reasonable output. However, the problem of model order selection and sensitivity to low signal to noise ratio still remains when LMedS is used. Note that in cases such as Figure 22, where LMedS fails to recover the correct parameters for the fit, the boundary between the inlier and outlier regions will be mislocated. A complete description and analysis of the edge detection algorithm is in preparation.

In Figure 26a a ring-on-steps range image is shown. Note the presence of both step and roof edges which are correctly delineated in the output of the edge detector (Figure 26b). Another example of range image edge detection is shown in Figures 27a and 27b.

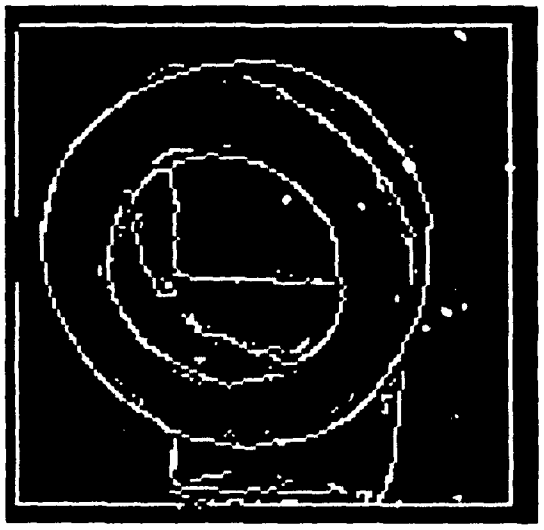
6 Discussion

The CBD algorithm uses decompositions in both the spatial and parameter domains. The spatial decomposition yields parameter distributions which are analyzed separately. Finding the extrema of a multidimensional function (optimization criterion) by proceeding along one variable (parameter) direction at a time is a frequently used numerical technique (see [11] for example). These techniques are known by different names: univariate search, coordinate descent, direction set, etc, and the decomposition is performed only in the space of the parameters. The condition for convergence and the influence of error propagation are well studied. In the next phase of our research we intend to apply the results obtained by the above mentioned numerical methods within the framework of the CBD estimator.

Recently Taylor [16] proposed the Cooperative Independent Parameter Spaces (CIPS) approach for improving the performance of the Hough transform in recon-



a

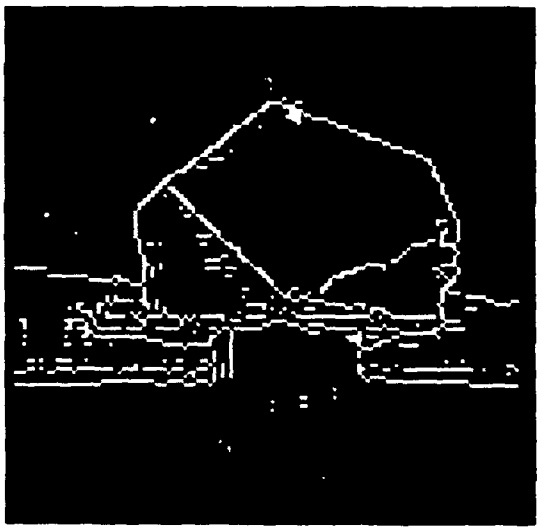


b

Figure 26: Simultaneous step/roof edge detection example. a) Ring-on-steps range image. b) Edge output.



a



b

Figure 27: Simultaneous step/roof edge detection example. a) Polyhedral object range image. b) Edge output.

structing range data with geometric models. In the CIPS approach, as in our CBD algorithm, the parameters are also analyzed in separate subspaces. The parameters are extracted with a multiwindow procedure, but fewer spatial subsets are used than in the CBD algorithm. The parameters are represented in discrete accumulators and considerable computational effort is required to reduce the artifacts of this quantization. In CBD the mode detection process uses the ordered parameter estimates sequence without quantization. The final estimate is obtained in CIPS by combining local estimates in a voting procedure. The CIPS approach and the CBD algorithm, while developed independently, have some common principles. However, the CIPS is tailored toward a specific application (object reconstruction) and several heuristic procedures are incorporated into it. The CBD, on the other hand, was derived from the class of high breakdown point robust estimators and makes extensive use of estimation theory concepts.

The CBD algorithm belongs to the *consensus vision* paradigm. This paradigm assumes that the data can be represented by a model of known structure but unknown parameters. The applicability of the consensus vision paradigm is contingent upon the existence of estimates which become unbiased whenever local agreement between the data and the model exists. The homogeneity conjecture states that the majority of the local estimates represent agreement with the model, i.e., the density of discontinuities in the data is not extremely high. Then a robust analysis of their global distributions can be used to obtain the final parameter estimates with increased accuracy, and/or to select the reliable local processes. In this paper our model was that of the piecewise polynomial surface structure and the locally LS estimated parameters are the unbiased carriers of the agreement with the model. For an application of the consensus vision paradigm to edge preserving image smoothing see [10].

The optimization criterion of the CBD algorithm is not that of least median squared residuals (12). The degree-0 LMedS in the enhanced mode seeking procedure minimizes the one-dimensional criterion (11) for each censored mean sequence separately. Thus the breakdown point of CBD is not expressed by (13) and is influenced by the reliability of the censoring process, i.e., by the signal-to-noise ratio. Our results show that in the range of normal SNR the CBD algorithm has a breakdown point close to the optimum 0.5. The absolute majority detected in the censored mean sequence may map only into a relative majority in the data (e.g., the largest set of points has less than 50 percent). Such cases appear when the data contains two discontinuities and thus we may have a "virtual" breakdown point for the CBD algorithm exceeding 0.5.

An estimator similar to least median of squares is least trimmed squares (LTS), in which the sum of the first h ordered squared residuals is minimized [13]. When h equals half the data size the breakdown point of LTS is close to 0.5. The presence of integration (the sum) in the LTS optimization criterion makes it numerically more stable and thus more suitable for applications than LMedS [14]. In the CBD algorithm, the degree-0 LMedS module could easily be replaced by the equivalent LTS estimator. Our experiments with LTS in the presence of zero mean noise, however, did not show a significantly better performance.

We conclude by recapitulating the advantages and disadvantages of the CBD estimator relative to LMedS. As advantages we can enumerate better tolerance of zero mean noise, less sensitivity to inadequate model order, and significant speed-up. The disadvantages are the requirement for larger processing windows when CBD is used in window operators, and possible artifacts caused by making decisions taken on the censored mean sequences instead of on the original data. Note that the disadvantages are of an implementational nature while the advantages are theoretical. This observation makes us believe that CBD can become a valuable tool in computer vision.

References

[1] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

[2] F.R. Hampel, E.M Ronchetti, P.J Rousseeuw and W.A Stahel. *Robust Statistics: An Approach Based on Influence Functions.* Jonn Wiley & Sons, New York, 1986.

[3] D.Y. Kim, J. Kim, P. Meer, D. Mintz and A. Rosenfeld. Robust computer vision: A least median of squares based approach. In *Proceedings of the DARPA Image Understanding Workshop*, pages 1117–1134, Palo Alto, CA, 1989.

[4] R. Kumar and A.R. Hanson. Robust estimation of camera location and orientation from noisy data having outliers. In *Proceedings of the IEEE Workshop on Interpretation of 3D Scenes*, pages 52–60, Austin, TX, November 1989

[5] P. Meer, D. Mintz, D.Y. Kim and A. Rosenfeld. Robust regression methods for computer vision: A review. *International Journal of Computer Vision*, 6:59–70, 1991.

[6] P. Meer, D. Mintz and A. Rosenfeld. Least median of squares based robust analysis of image structure. In *Proceedings of the DARPA Image Understanding Workshop*, pages 231–254, Pittsburgh, PA, September 1990.

[7] P. Meer and I. Weiss. Smoothed differentiation filters for images. In *Proceedings of the 10th International Conference on Pattern Recognition*, pages 474–479, Atlantic City, NJ, June 1990.

[8] D. Mintz, P. Meer and A. Rosenfeld. A fast, high breakdown point robust estimator for computer vision applications. In *Proceedings of the DARPA Image Understanding Workshop*, pages 255–258, Pittsburgh, PA, September 1990.

[9] D.C. Montgomery and E.A. Peck. *Introduction to Linear Regression Analysis.* John Wiley & Sons, New York, 1982.

[10] R.-H. Park and P. Meer. Multiresolution adaptive least squares smoothing of images. Technical Report CAR-TR-513, Center for Automation Research, University of Maryland, College Park, 199.

[11] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling. *Numerical Recipes.* Cambridge University Press, Cambridge, England, 1988.

[12] G. Roth and M.D. Levine. Segmentation of geometric signals using robust fitting. In *Proceedings of the 10th International Conference on Pattern Recognition*, pages 826–831, Atlantic City, NJ, June 1990.

[13] P.J. Rousseeuw and A.M. Leroy. *Robust Regression & Outlier Detection.* John Wiley & Sons, New York, 1987.

[14] P.J. Rousseeuw and B.C. van Zomeren. Unmasking multivariate outliers and leverage points. With commments by R.D. Cook and D.H. Hawkins; D. Ruppert and D.G. Simpson; P.J. Kempthorne and M.B. Mendel; and rejoinder. *Journal of the American Statistical Association,* 85:633–651, 1990.

[15] S.S. Sinha and B.G. Schunck. A robust method for surface reconstruction. In *Proceedings of the International Workshop on Robust Computer Vision*, pages 183–199, Seattle, WA, October 1990.

[16] R.W. Taylor. An efficient implementation of decomposable parameter spaces. In *Proceedings of the 10th International Conference on Pattern Recognition*, pages 613–619, Atlantic City, NJ, June 1990.

[17] A.P. Tirumalai, B.G. Schunck and R.C. Jain. Robust dynamic stereo for incremental disparity map refinement. In *Proceedings of the International Workshop on Robust Computer Vision*, pages 412–434, Seattle, WA, October 1990.

Correcting Chromatic Aberrations Using Image Warping

Terrance E. Boult
Center for Research in Intelligent Systems,
Columbia University Department of Computer Science, NYC, NY 10027.
and
George Wolberg
Department of Computer Science City College, CUNY, NYC NY.

Abstract

The problem of chromatic aberration arises because each wavelength of light is refracted differently by the elements of a lens. Unfortunately, this means that is image will be blurred and distorted. In color imaging these distortions cause measurable differences between the images. Recent research has proposed an approach for dealing with this aberrations by actively controlling the optics of the imaging system. This paper addresses the same problem, but instead of adapting the optics, we adapt the geometry of the (already obtained) images; we do chromatic aberration correction by image warping. We briefly discuss the image restoration/reconstruction techniques used, since they are non-standard. This is followed by a discussion of the techniques used to define the chromatic aberration correcting warp. The technique is demonstrated and analyzed on two test cases and is directly compared to the active optics approach.

1 Introduction

The first stage of an imaging system is the lens, which refracts the incoming light to focus it on the image plane. It has long been known that the refraction of light depends upon the wavelength, a ray refracted at the lens surface becomes a small spectral fan. According to geometric optics for a a simple lens, if a point object is placed in front of a lens, there would be a plane at some focal distance where the image of that point would be in focus, see Fig. 1. Unfortunately, because refraction is wavelength dependent, this focal distance is also wavelength dependent. The difference, due to wavelength, between the ideally focused image and actual image is called chromatic aberration. Chromatic aberration is generally broken up into two categories: axial aberrations and lateral aberrations [Slama *et al.*, 80]. In axial aberrations, the focal plane for one wavelength, say red, will be displaced along the optic axis from the focus plane for another wavelength, say blue. In lateral aberrations, the image of a one colored feature point will be displaced laterally within the image plane with respect to another color point. There are two parts to this lateral displacement. In general the largest component is a difference in the magnification, which causes a generally radial translation of image features. Secondly, there may be differences in the optic axis for the different wavelength.

The fact that some parts of a color image are more blurred that others, or that there the wavelength dependent distortions may not seem too important at first. If our algorithms were truly robust this might be true, but increased accuracy is always desirable, especially if it does not require expensive new equipment.

For example, consider a technique which requires clustering in color space. The effects of chromatic aberration can be very important, especially if the clustering makes assumptions on the shape/distribution of the cluster. Examine the color space depicted in figure 2 and imagine trying to find a "T" which is a process required in the color segmentation/highlight detection algorithm in [Klinker, 88]. Even if a color vision technique does not use the actual values of the RGB triples, but rather just uses

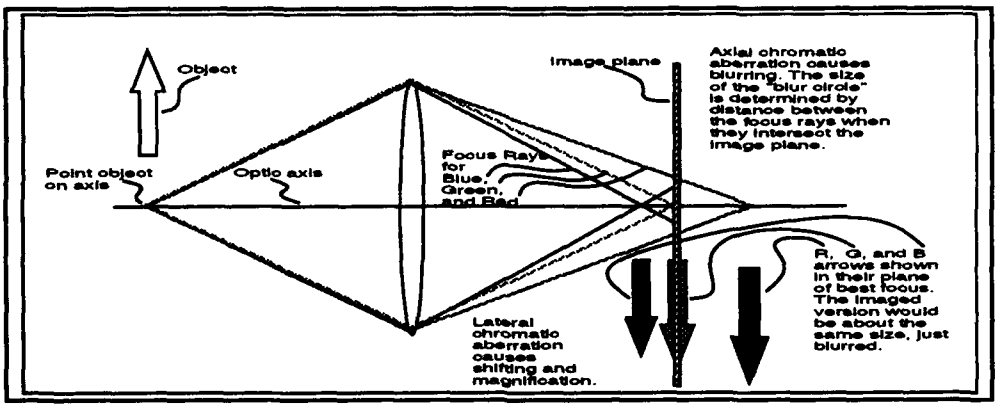


Figure 1: Figure showing the geometric optic interpretation for Chromatic aberration caused by a thin lens. Longer wavelength focus long, and are magnified greater than the shorter wavelength.

edges detected separately in each color band, the misregistration of one or two pixels may be significant.

Once one determines that chromatic aberration correction is necessary for a vision application, there are at least three things to do. First you can go out and buy an expensive lens and hope it is properly corrected.* We briefly discuss, in the next subsection, some of the techniques of lens designers.

A more recent development is the use of active lens control to achieve reduced chromatic effects. This technique, developed by R. Willson and S. Shafer at CMU [Willson and Shafer, 91a] and [Willson and Shafer, 91b], takes three separate images with slightly different focus and zoom settings designed to compensate for the optics. This is discussed in section 1.2.

The final choice is to do image warping for chromatic correction, as described in this paper. We will show, in section 2.2 how to determine the warping function and briefly recall how to do the warping and the image restoration/reconstruction needed for it. Then we will demonstrate and analyze the algorithm in section 3.2.

We are not the first researchers to suggest using image warping for image registration or correction, e.g. NASA has used image warping in various applications, see [Green, 89], and much of the early work on image reconstruction centered around "digital correction", e.g. see [Rifman and McKinnon, 74]. We are unaware, however, of any quantitative experimentation studying the effectiveness of warping to correct for chromatic aberration.

1.1 Chromatic aberration correction by better lens design

We now briefly discuss the "traditional" techniques for dealing with chromatic aberration. This reivew of lens design is based on the material in [Slama *et al.*, 80], and [Kingslake, 78]. The subject of design good lenses is quite involved and increasing uses computer simulations. While one might think the space of lens designs have been completely explored, modern techniques continue to produce better lenses, see [Laikin, 91]. If the price is right, a lens can be designed to meet particular imaging criterion. However, most vision researchers use off the shelf lenses with either no chromatic correction or only simple correction. In addition, the correction of aberrations is generally becomes more difficult as one reduces the focal length, increases the aperture, increases the field of view, or allows zooming. Note that wide field of view, large aperture zoom lenses, with short minimal focal length, are exactly what many vision researchers use.

Axial chromatic aberration is usually handled by using multiple elements, some positive others negative, with different optical indices. These elements are chosen such that if the first causes the longer wavelength (red) to focus too far, the next causes the shorter wavelength ((blues) to

*In [Willson and Shafer, 91b] they report that they have tested a number of lenses, including some that are supposed to be chromatic correcting lenses, only to find significant errors in every lens.

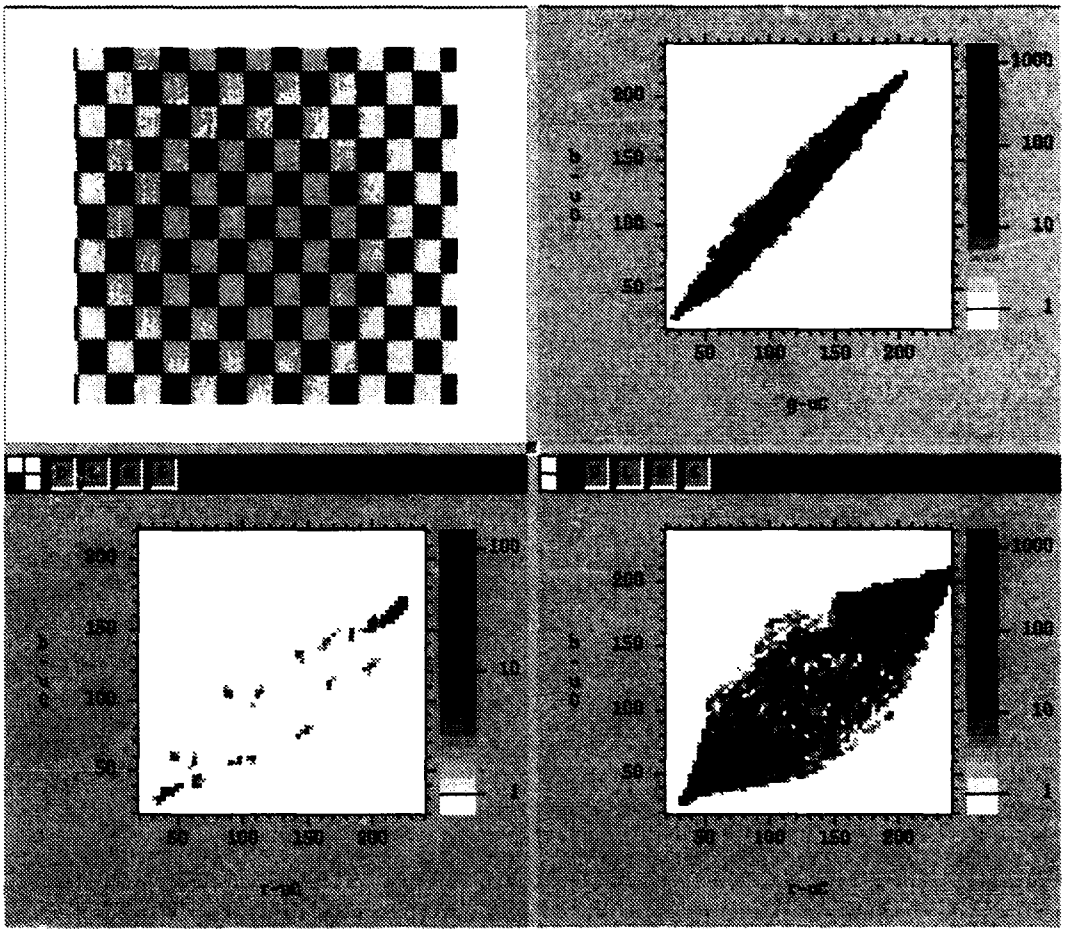


Figure 2: Upper left shows a test image (the blue channel). The figures on the right show 2D histograms of color space top is blue vs green, bottom is red vs green. If there were no chromatic aberration both of these plots would be a line. The original images were taken when the lens was focused for blue light. Thus for this uncorrected lens system, the green channel show some errors, and red channel much larger errors. The graph in the lower left is the plot of red vs blue for a square window in the lower left of the image. These plots are described in more detail in section 3.

focus long. A standard item, called an achromatic doublet, uses two-lenses made with different glasses such that it can bring 2 wavelengths into alignment on the axis. When there are more elements in the system, more wavelengths can be brought into axial alignment. For the simple "corrected" lenses two wavelength are brought into correspondence resulting in a quadratic like error between these wavelength. In such lenses, deep violets and deep reds generally focus short and the middle wavelength, like yellow, focus long, see [Slama *et al.*, 80], [Kingslake, 78]. Note that this correction is generally measured on the optic axis, and in simple systems degrades with increasing distance from the axis. Many "three lens" systems correct on the axis and in a zonal band some distance from the optic axis producing a spatially varying correction. There is often a tradeoff between correction of chromatic aberration and correction of shperical aberrations and commas. Still high quality photogrametric lenses can be obtained with axial chromatic errors of less than .01%.

For simple lenses the correction for lateral aberration is, in theory, easier. To correct for the first order effects, all that is necessary is for the prismatic effects of the first lens to be canceled in the second. For simple symmetric lens designs this is straightforward, see [Slama *et al.*, 80]. Unfortunately, for asymmetric lenses, e.g.

zoom and tele-photo lenses, the correction is considerably more difficult (some of these have 15+ lenses with complex internal moving parts). Additionally, if the lens has higher order lens affects, i.e. comma, radial and tangential geometric distortions, these are also wavelength dependent complicating the correction process.

While the "older" lenses were often designed by hand and as a result corrected only for a few wavelength at a few points, modern computational techniques have allowed lens designers the ability to numerically designs lenses which have minimal aberration. Actually manufacturing and maintaining such compensation is, however, another issue. With increasing part number the chance of misalignment of an internal lens (or shifting of the lens after manufacturing greatly increases. Furthermore, it is very common in modern lenses to use coatings on the optics to reduce reflections. Such coatings are also wavelength dependent and good systems use a multi-coating technique to reduce the wavelength dependence and decrease chromatic aberrations. Such coatings are, however, easily damaged or removed severely impacting the performance of the lens.

1.2 Chromatic aberration correction by active control

Because of the emerging use of color imaging in computer vision, and because of the expense of well corrected optics (which are still not perfect), researchers at CMU have recently been investigating the use of active lenses elements to deal with chromatic aberration, see [Willson and Shafer, 91a], [Willson and Shafer, 91b]. We describe their approach in some detail since our experiments will be compared with their approach and on their data.

The CMU active optics approach to chromatic aberration correction has three main steps:

1. determination of best focus for each color,
2. determination of a magnification factor for red and green,
3. and determination of camera shift to align images.

The first stage corrects for axial chromatic aberration by varying the focal plane for each of the colors. Note that for images with only three wavelength, this can exactly correct for axial errors. For more complex images this will have zero error for three wavelength and, hopefully, small error for the remaining wavelengths. This approach can be applied even if the lens was chromatically corrected. The technique is automated and determines best focus using an algorithm by E. Krotkov, see [Krotkov, 87]. This involves computing an image sharpness measure at a number of focal positions and searching for the maximum of this sharpness measure.

The second stage begins to correct for lateral chromatic aberration. They determine a magnification factor for each band and use this to actively control the zoom lens. Unlike focus determination, this stage requires more than a simple image based operator; it requires some type of geometric calibration image. They have used subpixel detection of edges (vertical and horizontal) to determine this zoom. They then compute the magnification needed and actively update the lens for each of the green and red images.†

The final stage is to deal with differences in the image of the optic axis. Because the lens elements are not all perfectly aligned, it will often occur that the image of the red optic axis is displaced with respect to the blue optic axis. When this difference is coupled with refocus and zooming, the effect can become more pronounced. Thus using the above mentioned edge data they also compute a shift for red and green to bring them into correspondence with blue. This is done by physically shifting the camera in the plane of the imaging sensor. Obviously, because of the magnification affects of the lens very small shifts, on the order of .005in, will generally be used.

We feel it is worth noting that their paper also addresses use of active optics in shape from focus computations. In this case, the active optics for chromatic correction is a minimal incremental cost in that of shape from focus. Because they also correct for axial aberrations (by refocusing), they can, theoretically, correct for

†Blue is assumed as the correct image. Their errors might actually be smaller than reported in their paper had they chosen green to be the standard and actively corrected both red and blue.

blurs that would be problematic in shape-from-focus when used on a colored world.

2 Correction Using Image Warping

When we did our initial work on imaging-consistent reconstruction algorithms, [Boult and Wolberg, 91], we realized that one use of such techniques would be in image warping to correct for lens aberrations. However, at the time we felt that the cost of of image warping for lens correction was probably unwarrented since we assumed (from reading [Slama *et al.*, 80]) that the available lenses probably did not have significant chromatic aberrations and that radial type lens distortions could be handled by directly mapping the location of feature points as opposed to warping the intensity image before feature detection. When we heard a talk by Reg Willson, about the CMU active lens approach to chromatic aberration, we realized we were wrong; standard CCTV lenses have noticeable aberrations that require correction. Furthermore, an increasing number of "physics-based" vision algorithms use the actual radiometric quantities measured, and hence they need more than just a calibration of the geometric distortions, they need to have the intensities registered. Unlike human interpretation, these phsyics-based algorithms are probably not particularly robust with respect to violations of their imaging assumptions.

There are two main parts of the image warping technique to chromatic aberration correction: how to warp images in general, and determining what warp to apply. We will briefly discuss each of these.

2.1 Image warping

Image warping has been most commonly used in graphics, where the underlying question is "does it look good", a qualitative assessment. To use image warping in vision we need to address the question, are the pixel values correct, a more quantitative question.

For complex warps we described a technique to increase the accuracy of the warp while maintaining low cost, see [Wolberg and Boult, 89]. This separable image warping technique warped the image twice and then combined the results to increase the accuracy. Fortunately, the warps needed to correct for most lens distortions are not so severe and do not require such a complex algorithm. Instead, ignoring boundary effects, we can simply warp the image in one direction (say x) and then the other direction (y). This is accomplished row by row and then column by column.[‡] This allows for both efficient pipelining (within a scan) and parallelism (each row is warped independently).

This leaves us with the question of how to warp a regularly sampled 1D signal into another 1D array of pixels, with a nonlinear transformation between their geometry. To do this we need some approximation to the signal underlying the input 1D signal, and some way of warping this signal. We approximate the signal with what we call imaging-consistent reconstruction / restoration filters. These linear filters use a model of input PSF (blur) *within a pixel* to obtain a functional restoration. This functional form is then then warped, and reblurred according to an output PSF, using an approach we call the integrating resampler. An important property of these filters is that they do not necessarily pass through the data, but rather when blurred according to the assumed input PSF they return the original data. While there are many variations on this idea, this paper mainly considers one, called a quadratic restoration filter with a rect type PSF, or quadratic-box restoration for short. We define the function here, but see [Boult and Wolberg, 91] for more details and discussion. This filter is local, so consider pixel i with which spans the interval from k_i to k_{i+1}. We assume some interpolation technique, e.g. linear interpolation or cubic-convolution ([Rifman and McKinnon, 74]), is use to determine values e_j at k_j. The value of the quadratic-box restoration function is then given by:

$$e_i + (6v_i - 2e_{i+1} - 4e_i)x + 3(e_{i+1} + e_i - 2v_i)x^2$$

where $x = \frac{t-k_i}{m}$. The integral of this quadratic over the interval k_i to k_{i+1} is exactly v_i the measured input pixel value. In this paper we consider the use of linear interpolation as a means

[‡] Actually for increased accuracy we might use full 2D filtering, but this would really push up the cost of the algorithm.

of determining the endpoints because its cheap. We have also determined the endpoints using cubic-convolution with numerous values of of their magic parameter A, with no significant visible difference in this application.

Given the "restoration" of the input function, we now have a functional form to warp, which we do using the following idea we call the integrating resampler. Assume the n input pixels, at locations $k_i = \frac{i}{n}, i = 1..n$, are being mapped into m output pixels $o_j, j = 1..m$, according to a warping function $g(t)$. Compute q_j as the linear approximation to the location of $g^{-1}(o_j)$, i.e.

$$
\begin{aligned}
&for(j = i = 0;\ \ j < n; j{+}{+})\ \{ \\
&\qquad while(g(k_i) < j)\ i{+}{+}; \\
&\qquad q_j = i + \tfrac{j - g(k_i)}{g(k_{i+1}) - g(k_i)} \\
&\}
\end{aligned}
$$

Now to process the data we run along the input determining the next event: either an input pixel will be consumed, or an output pixel will be generated. If the next event will be the completion of an input pixel we compute the integral from the location of the last event to the end of this input pixel, adding the value to the accumulator. If, however, the next event is the generation of output pixel j, we use q_j as an approximation of $g^{-1}(o_j)$, which gives the location of this event in the input space. We then compute the integral from the location of the last event to the location of this event, and add this value to the accumulator.** For more details on this process see [Wolberg and Boult, 91]. The underlying idea of this integrating resampler can be found in [Fant, 86] which proposed a similar algorithm for the special case of warping a linear reconstruction of the input.

As mentioned before, other researchers have used image warping in vision. In particular, researchers at NASA have frequently used image warping, though more often to allow better human viewing than for the application of vision algorithms. A standard approach is described in [Green, 89]. This approach uses a very simple reconstruction filter (bi-linear interpolation) with point sampling. While an example presented, there is no quantitative analysis of the approach.

**In the general case we would compute the integral with the output PSF, *in this* case a rect filter.

2.2 Computing the distortion

As mentioned previously, there are really two related types of chromatic aberration, axial and lateral. The main goal of image warping-based chromatic aberration correction is to correct for the lateral aberrations. To do this we start with the the same types of geometric features used to compute the zoom and shift factors in the in the active lens approach. In this case we have data on location of horizontal and vertical edges in each of the R, G, and B images (with the blue image focused). These are the same uncorrected images used in [Willson and Shafer, 91b], and we use the edge data computed by their algorithms.

The test target was a checker-board pattern, see Fig. 2. For each of the sides of each checker, the horizontal edge position provides a stable measurement. Since the feature detector only computes the horizontal edge position near a vertex, each horizontal edge position has an accompanying vertical position which, while not stable enough to use in warping, facilitated grouping of related horizontal edges. Edges with approximately the same vertical location were thus associated to form a "row" of horizontal positions in each color band.††. A similar matching was done to form "columns" of vertical edge position data, using the tops and bottoms of each checker. For the CCTV example below we obtained 270 horizontal edges points and 416 vertical points. For the Photometrics example there were 210 horizontal edge points and 220 vertical edge points. We do this processing for each of the three color bands. Using the edges in the blue image as the desired location, we the difference between those positions and the computed edge positions for red (or green) to define the warp. To find the warping at other points we need to fit some model. To maintain flexibility, we choose to fit a cubic spline through the location data in each row/column and consider the

††Unfortunately, every row did not have the same number of detected horizontal edge features. To make our computation easier, we used linear interpolation between the data points with minimal variation to get an equal number of datapoints per row

warp to be the tensor product of these splines.

Once we have computed the warping function, we can then use the integrating resampler, as described in section 2.1 and [Wolberg and Boult, 91], to warp the red channel and the green channel to their respective desired positions. Note that other techniques might be used to determine the warping function. In particular, a future direction will be to consider a global deformation model, which might be parameterized by the lens setting for focus and zoom. It will also examine warping all three images to correct for geometric lens distortions as well as chromatic aberration. We note that in [Green, 89], they used correlation between features in each color channel, as well as *a priori* calibration information. Again, however, the results seemed to intended for human analysis and no quantitative analysis was given. Future work will compare use of correlation with edge based warp determination.

3 Experimental Analysis

In this section section, we discuss the results of application of the image-warping chromatic aberration correction technique on two test images. The test images were obtained from the Calibrated Imaging Laboratory at CMU, and were used in [Willson and Shafer, 91b] to describe/test the active optics approach to chromatic aberration correction. We first discuss how to measure the quality of a correction, then get into the actual data.

3.0.1 Photometrics examples

These images were also collected at CMU, and used a Photometrics camera connected to Matrox frame grabber. The lens was a Fujinon motorized zoom lens. For the color filters they used a Hoya IR block + the same RGB filters. The 1/2" checkerboard was imaged at a distance of 2.03m. The original image is not shown. The actual sensor array (384x576) is larger than the lens's spot size, and so the images were clipped to 338 by 388. For each image 10 frames were averaged together. The integer pixel values (0..4059) were converted to the range (0..255). The color histograms of the uncorrected data are shown in Figs 7.

3.1 Determining the quality of a correction and displaying the results.

In their paper, [Willson and Shafer, 91b], Willson and shafer use the displacement of the location of the zero-crossings edges as an error measure. In this paper we do not use such geometric measures but rather use colormetric measures. The primary reason for this change in error measures is that we will directly manipulate the geometry of the image and felt this would be an unfair measure to apply to our algorithm. Secondly, the edge position does not actually address the issue of focus in each band since a blurry red image might have its edge in the same position as blue but the color properties of the images could be poor.

Since the actual scene is a black and white checker pattern, all pixels in the scene should image to some shade of gray. To visualize the errors we consider two techniques: direct plotting, and computing an error measure. The first, direct display, involves plotting two 2D histograms where for each pixel its b pixel value is used as the y coordinate of the plot while the x coordinate is the r pixel value or the g pixel value. In these plots, their is some ideal curve for the camera response. If the camera is linear the ideal would be a straight line. The important information in these plots is the spread of the points around their "central" tendency, the wider the spread, the more uncorrected the images.

Examples of these plots can be seen in Fig. 2 which shows the 2d histograms for the uncorrected images for the CCTV camera example. Given the intensity ranges of the images we show the part of the histogram containing data. On the right of that figure are the plots for green vs-blue (top) and red-vs-blue for what we call the big window. Because we did not have information about the warp at the very edges and because without a global model to allow extrapolation of the warping function we consider only the window with rows [15, 465] and columns [15, 497]. Note that the active optics approach also has its poorest performance near the boundary, and all comparisons will consider its performance in tne same window. The 2D histograms for the big window uses intensity to encode the

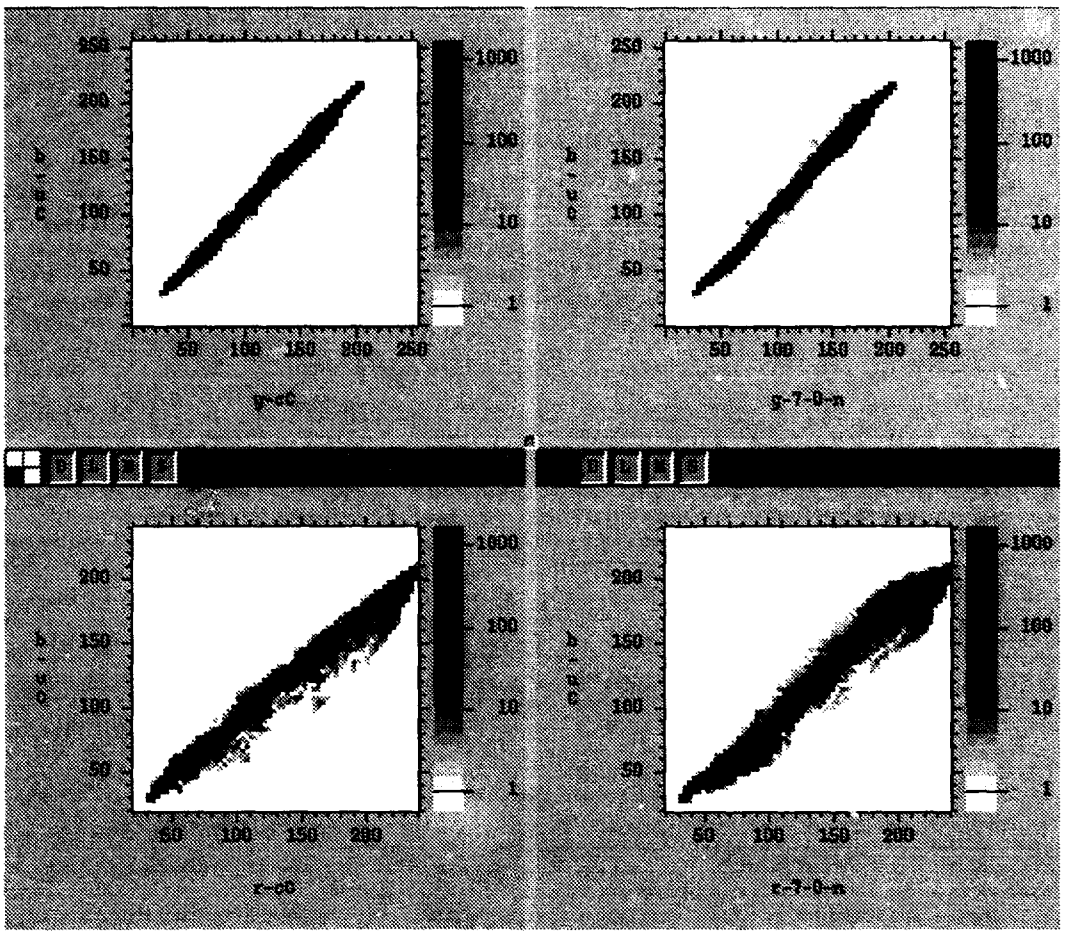


Figure 3: 2D histograms showing dispersion in color space using the big window [15 465] × [15 497]. The top row shows the corrections to green (blue-green histogram), and the bottom shows corrections for red. The left shows the results obtainable with active optics. The right shows the image warping results. Both techniques work extremely well on green and do a credible job on red, for quantitative comparisons see table 1. Overall the active optics approach is better (tighter cluster) and also more symmetric in its error. The sigmoid shape which is slightly visible in the blue-red histogram for image warping is caused by differential blur after correcting for magnification affects.

logarithm of the number of items in each of the 64x64 bins, with black meaning 1000 pixels in that bin. Note that a few bins are clipped because they contain > 1000 pixels. On the left of Fig. 2, we see the 2D histograms for the red components, this time restricted to a window in the lower left of the image, [15 65] × [15 65]. This represents a region which is, approximately, maximally distant from the optic axis and hence where one would expect to find maximal artifacts. The plots are similar in nature except that they use a log scale with 100 items being black.

The remaining measures we will report are quantitative. While it seems intuitive to consider RMS distance to the gray-line, this has a problem: the image values are not normalized, and we did not have radiometric calibration information. Thus we adopted the following approach. Because the only information is near the edges we restricted our attention to this region.[‡‡] The rows in tables 1 and 3 marked "outside mask" computed their values outside this mask region to show the size of the error

[‡‡]Determined as an expansion of the region where the gradient magnitude was > 4 for the CCTV example and > 2 for the Photometrics examples. (Obtained by the KBVision sequence FastGauss(2), GradIm, NormIm , ThrshIm followed by a MorphOps with "d8 5e8".

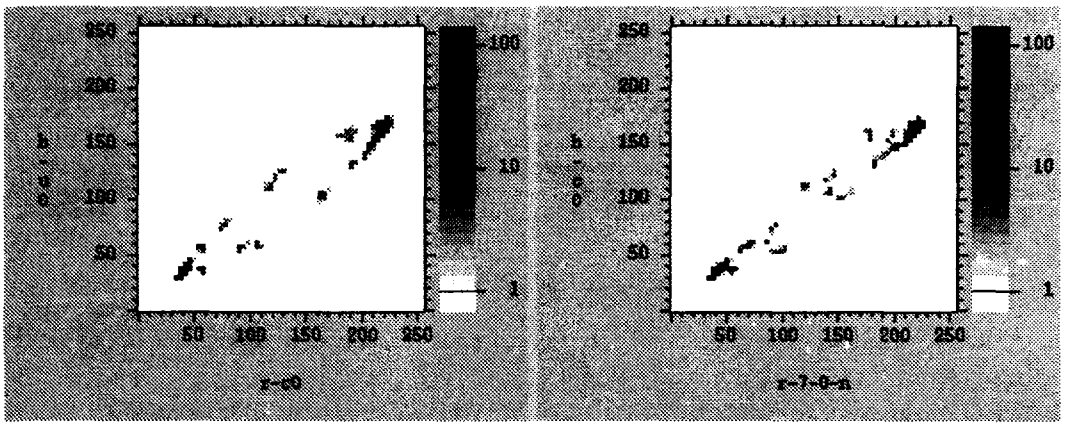


Figure 4: 2D histograms showing dispersion in color space using different windows. The shows the results on the box from the lower left of the image, [15 65] × [15 65]. In this case the active optics (left) does not do as well, probably because of geometric lens distortions affecting the image. (The open nature of the plot is indicative of a magnification error.) The image warping approach, lower right, does reasonably well. It too has more problems in the corners, possibly because of inaccuracies in the warping, plus focus affects. These observations are also supported by the quantitative results in table 1.

Algorithm	Region	Gray-line error	BW-RGB error	BW-R error	BW-G error	BW-B error
Uncorrected	outside mask	237.234	0.096	0.073	0.069	0.076
Uncorrected	[15 465] [15 497]	568.381	0.363	0.300	0.261	0.260
Active Optics	[15 465] [15 497]	378.378	0.353	0.284	0.257	0.260
Image Warping	[15 465] [15 497]	411.030	0.364	0.301	0.261	0.260
Uncorrected	[15 65] [15 65]	616.824	1.049	0.861	0.761	0.756
Active Optics	[15 65] [15 65]	557.978	1.039	0.838	0.765	0.756
Image Warping	[15 65] [15 65]	449.579	1.053	0.869	0.760	0.756
Uncorrected	[50 400] [270 300]	778.941	0.880	0.692	0.636	0.673
Active Optics	[50 400] [270 300]	637.172	0.875	0.681	0.637	0.673
Image Warping	[50 400] [270 300]	703.478	0.880	0.693	0.635	0.673
Uncorrected	[235 260] [50 430]	554.214	0.831	0.692	0.594	0.592
Active Optics	[235 260] [50 430]	363.741	0.800	0.641	0.585	0.592
Image warping	[235 260] [50 430]	472.019	0.833	0.696	0.595	0.592

Table 1: Table of error for CCTV examples. As can be seen the active optics approach produces quantitatively better values for all examples except the one in the lower left part of the image. Note the differences in the blur related measures on the vertical vs horizontal windows in the images.

measures outside this region of interest.

Given that we do not have radiometric calibration information, we use a simple heuristic to compute approximate information. We know that the calibration target was to be black and white. We computed a set of reference values for every 10 x 10 window in the input. These references values are obtained by considering only pixels outside the mask described above but within a window of 60x80 pixels for the CCTV (25 x 60 for the Photometrics) centered around the current point. Within this window we consider average all those above 130 to get a "white" level and all those below 100 to get a "black" level. This was done separately for each color band. Using these reference values, we define five error measures. The first, which we call *Gray-line error*, is the average distance from a RGB triple to the line defined by the reference values. This error measure relates to the color shift of a pixel. The remaining measures were meant to be more sensitive to blur

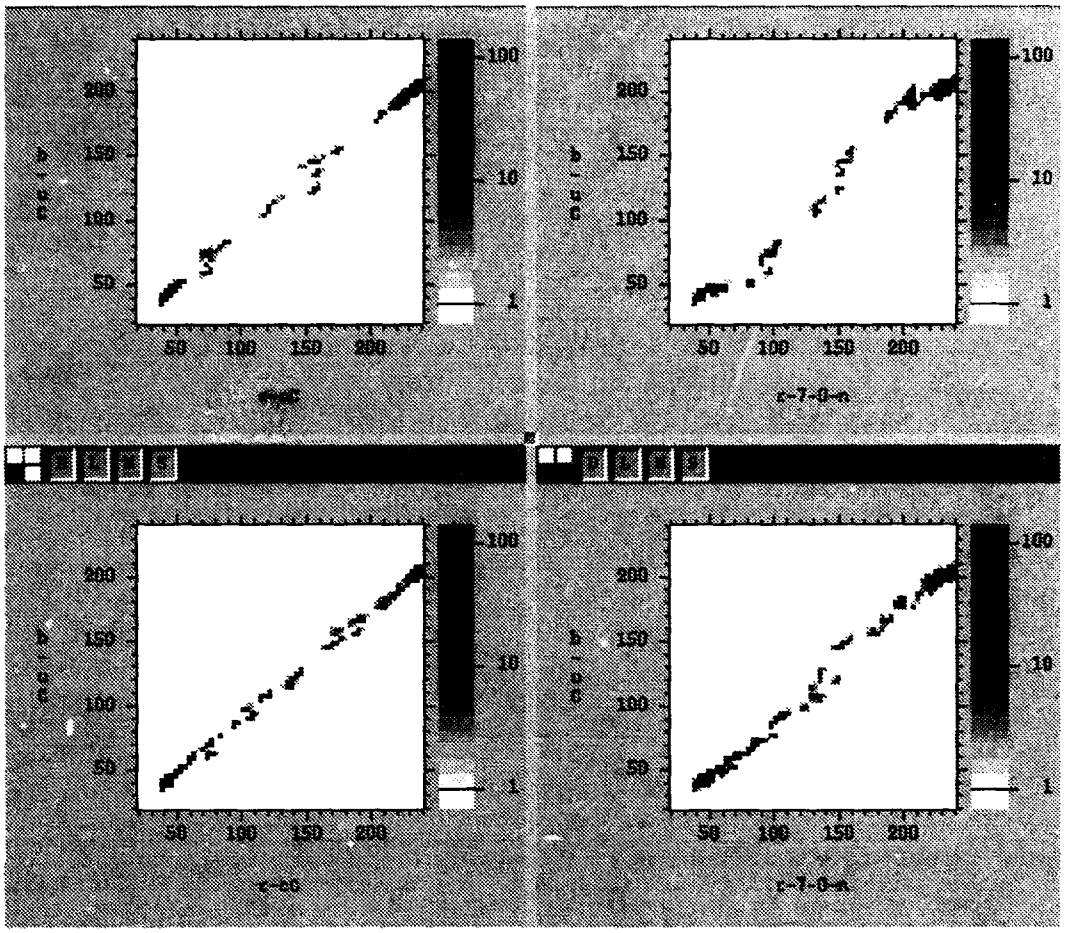


Figure 5: 2D histograms showing dispersion in color space using only edges of one type. The upper plots show the performance of the two algorithms (active optics on the left) when applied to vertical window with extents [50, 400] × [270, 300]. This window contains only horizontal edges. In this case the active optics does extremely well (it is best on or near axis). As can be seen in the the upper right the image-warping technique has a sigmoid shaped bias. This is caused by difference in focus between the red and blue channels. The bottom row uses the horizontal box [235, 260] × [15, 430] which contains only vertical edges. Again, active optics is right on the money. The image-warping technique does not show the differential blur as in the vertical case, but this time shows some residual magnification error (mostly near the outer edges of the image.) We are not sure about the exact cause of this directionally selective behavior, but have found a similar difference in the blur of uncorrected images.

within the image. Note that the *Gray-line error* could be made small by overly blurring the image, because everything would approach gray! The second quantitative error measure, which we call *BW-RGB error* is defined as the mean pointwise distance from each RGB triple to the nearer of the reference triples. Since the underlying image was supposed to be black and white this measure should be small. If the images are locally blurred this measure will grow. If there is excessive blurring, the heuristic calibration processes will cause this measure to become too small. The remaining measures were meant to be sensitive to blur separate in each wavelength. The third measure, which we call *BW-R error*, is the distance between the R value of a pixel and closer of the reference values for R. The measures *BW-G error* and *BW-B error* are defined similarly.

Under ideal imaging, each of the error mea-

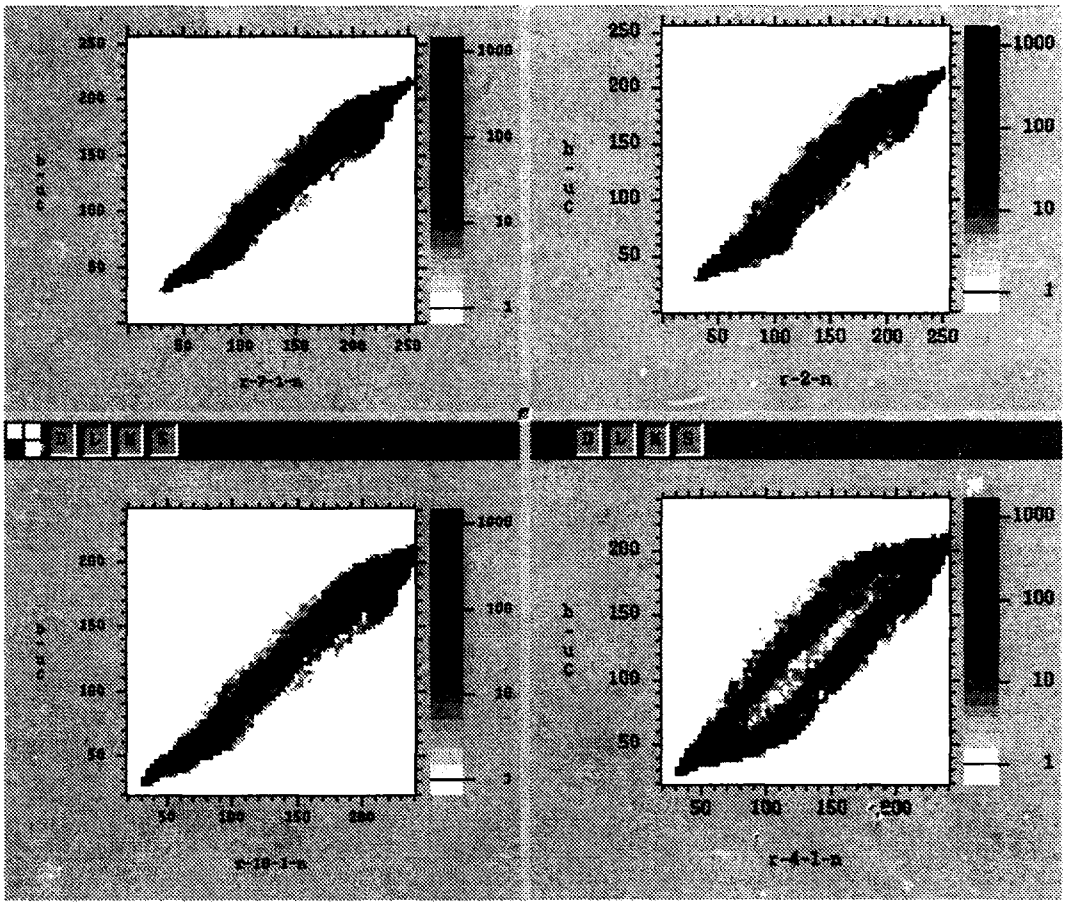


Figure 6: 2D histograms showing dispersion in color space using the big window and the red channel. Here we see the 4 different image warping results using different image reconstruction techniques. The upper left is the quadratic-box using cubic-convolution based edge values with a value of $A = -1$. This results in a correction which is barely different from using linear interpolation to get the edges. The upper right shows the results using just linear interpolation for image reconstruction. While it still does well, it is not as tight a cluster as the quadratic-box restoration approach. In the lower right we see the results of applying the integrating resampler using cubic-convolution with $A = -1$. While it may be a superior image reconstruction filter, it has significant problems in this application. We also tried cubic-convolution as it was originally intended (i.e. doing point sampling reconstruction), and the results were even worse. On the lower left we show the result of another restoration filter. This one is also a quadratic spline, but the point spread function used to define it was a 4 piece cubic approximation to a Gaussian.

sures would be zero, and for non-ideal imaging smaller error measures are better. Unfortunately, interpretation of these measures is complicated by the fact that they do not have intuitive units of measure, so its not clear how important a difference of 10 units is in the gray-line error. Additionally, because the images have noise and also because our approximated calibration data is not perfect, the error measures have an offset so that it is unlikely they would ever attain the value zero. When we report the quantitative results, we will also report the error measures in the area outside the aforementioned edge mask. In these regions the image should have little chromatic aberration, and the residual error measure gives some indication of the base level of each error measures.

3.2 Analysis of results

The analysis is mostly the data itself, presented in figures 3–8 and tables 1–3. The cap-

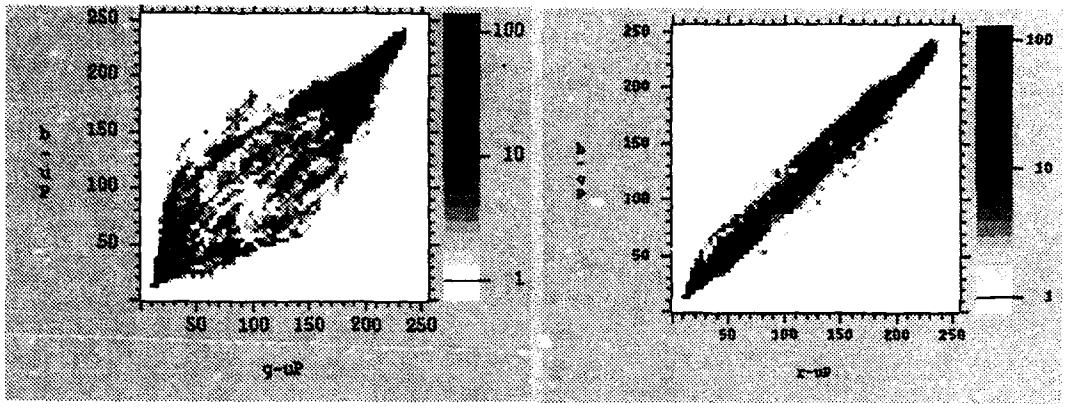


Figure 7: Here we see some examples using a Photometrics camera and a Fujinon lens. The figure shows 2D histograms of color space, with blue vs green on theleft, and red vs blue on the right. The lens was supposed to have been corrected for chromatic aberration. Obviously it was not completely corrected. Note that these plots are on a different scale (black =100) than the CCTV examples (black=1000).

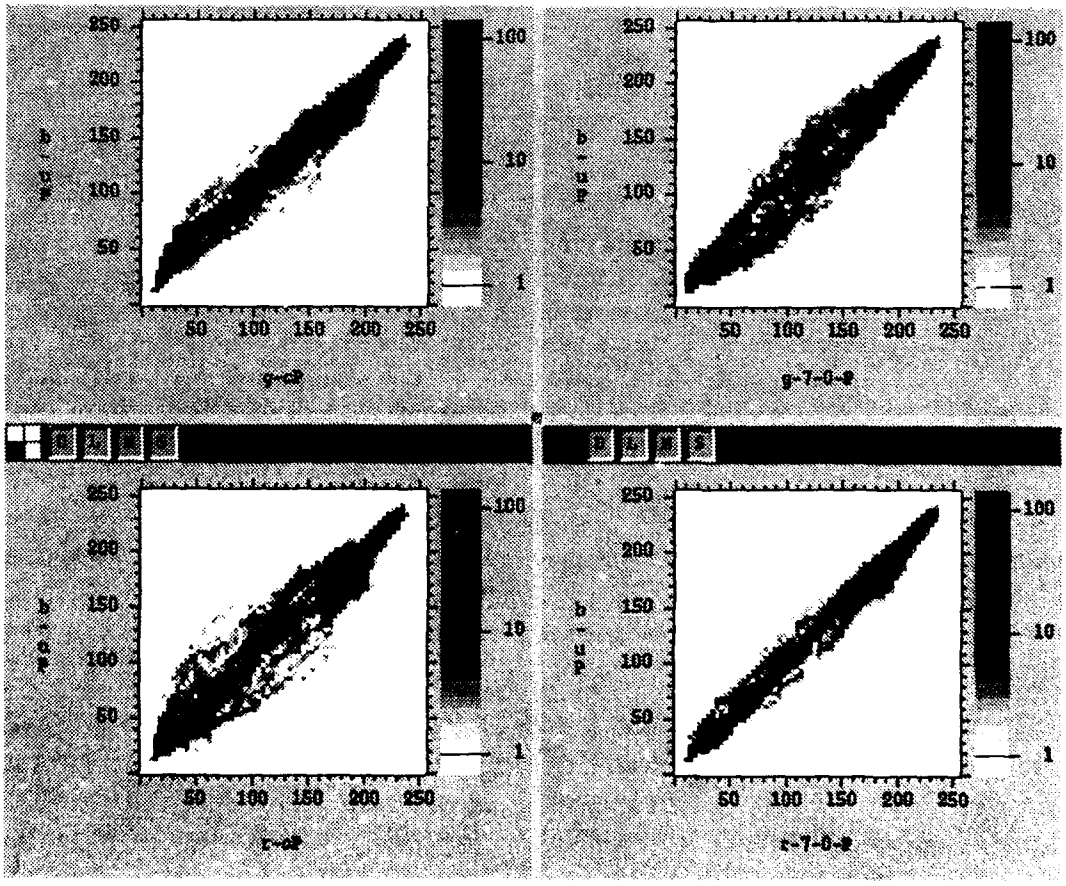


Figure 8: Here we see the corrected versions of the Photometrics example. Green is on top, red on the bottom. On the left are the active optics approach. On the right is the imaging warping approach. Qualitatively, we did comparably well, for a quantitative comparison see table 3 While the active optics approach reduced the RMS error for the red channel, it did increase the size of the envelope in color space.

Algorithm	Gray-line error	BW-RGB error	BW-R error	BW-G error	BW-B error
Uncorrected	568.381	0.363	0.300	0.261	0.260
Active Optics	378.378	0.353	0.284	0.257	0.260
Quadratic-box Restoration, edges from linear interpolation	411.030	0.364	0.301	0.261	0.260
Quadratic-box Restoration, edges from Cubic Convolution, A=-1	412.021	0.363	0.301	0.261	0.260
Quadratic-gauss restoration, edges from Cubic Convolution, A=-1	411.744	0.364	0.301	0.261	0.260
Bi-linear Interpolation	453.921	0.366	0.303	0.264	0.260
Cubic Convolution, A=0	581.295	0.367	0.305	0.266	0.260
Cubic Convolution, A=-1	585.868	0.363	0.300	0.260	0.260

Table 2: Table of error for different reconstruction algorithms applied to CCTV example. There is a slight sharpening when using $A = -1$, but because the warp is rather small, the sharpening is not very significant. Note that the difference between the new reconstruction methods and linear interpolation is about the same as the difference between active optics and the new methods. Finally, cubic convolution seems worse than the uncorrected image, although the qualitative results looked like some improvement. We are still investigating this behavior.

Algorithm	Region	Gray-line error	BW-RGB error	BW-R error	BW-G error	BW-B error
Uncorrected	[outside mask]	306.499	0.154	0.113	0.119	0.119
Uncorrected	[15 323] [15 323]	694.093	0.391	0.302	0.288	0.301
Active Optics	[15 323] [15 323]	456.365	0.392	0.301	0.291	0.301
Image Warping	[15 323] [15 323]	562.715	0.399	0.304	0.305	0.301
Uncorrected	[15 65] [15 65]	805.603	0.969	0.731	0.670	0.791
Active Optics	[15 65] [15 65]	546.352	0.968	0.709	0.694	0.791
Image Warping	[15 65] [15 65]	567.845	1.027	0.774	0.775	0.791

Table 3: Table of error for Photometrics example

tions in the figures are meant to be almost self contained, and present most of the commentary. The 2D histograms are meant to provide a qualitative measure of performance, and clearly show the largest errors. The tables provide a more quantitative comparison with a unweighted sum of distances type mixing of large and small errors. The references to the active optics approach referees to [Willson and Shafer, 91b], who were kind enough to supply us with their data. References to quadratic-box restoration refer to the method described in section 2.1, [Boult and Wolberg, 91] and [Wolberg and Boult, 91], with edge values determined with linear interpolation. Details on cubic convolution, which is use in figure 6 and table 2 can be found in the 2 previous references as well as [Rifman and McKinnon, 74] and [Park and Schowengerdt, 83]. We note that the figures use cubic convolution in the integrating resampling approach. We also tested cubic convolution using point sampling, and its performance was slightly worse.

3.2.1 CCTV camera based examples

These images were collected at CMU, and used in [Willson and Shafer, 91b]. The used a General Imaging camera connected to Matrox frame grabber. The lens was a Cosmicar motorized zoom lens (12.5-75mm), with a minimum focus distance of 1.2m. For imaging they used a Corion IR block + filters for R,G and B.*** The the 1/2" checkerboard was imaged at a distance of 1.5m. The original blue channel and initial color histograms were shown in Figs 2.

3.3 Improvement to the image warping technique

If the original images had been taken at a point of maximal focus for yellow (or "white"),

***The used Wratten filters. #25 + 0.9ND for red, #58 + 0.6ND for green, and #47B for blue.

then the results of the image warping techniques would probably have been even better. A significant part of our error is due to the differential blur between wavelengths, which is not corrected by warping. In the CCTV case this difference should be maximal when focused on an extremal wavelength (e.g. blue) as was the case here. Note that focusing on yellow or an unfiltered image should reduce the error in the uncorrected images as well, but should not affect the results of the active optics approach.

The image warping technique would also likely benefit from a denser set of feature points, especially near the edges where the lens distortions are changing most quickly. Future work will examine the calibration techniques to correct both for chromatic aberration and radial/tangential lens distortion. Further we will be looking into determining a functional form, parameterized by zoom, focus and aperture settings, for the correction functions.

As discussed below, one of the main advantages of the active optics approach is the ability to correct for axial aberrations, i.e. wavelength dependent blur. An interesting approach would be to use active focus for each channel, then use image warping on the results to correct for lateral chromatic aberration. This should provide a good tradeoff between fidelity and cost of implementation since it would require only active focus control. In fact, the focus differences might be implemented directly in an RGB sensor, e.g. by beam splitting and imaging each separately with a slightly different focal length.

4 Critical Comparison

We now critically compare the two techniques, in a relative fashion. We show a + when the image warping technique has the advantage, a − when the active optics approach has the advantage, and a ±, when the advantage might depend on the application.

+ Warping can be applied to images taken with an "RGB" camera where each frame is collected simultaneously. Thus it has the potential for use in color sequences.

+ Warping does not require specialized equipment. While the use of motorized zoom/focus is growing, the ability to precisely translate the camera between images is less common.*

+ Warping can handle significant chromatically varying geometric distortions even if no focus/zoom could correct for them (such as higher order lens distortions).

+ The warping approach holds the potential to correct for other geometric and radiometric aberrations at the same time it corrects for chromatic aberrations. This, however, would need more calibration information.

± The warping approach can be successfully applied to lenses that have undergone "chromatic aberration correction". Because of the complex nature of the chromatic aberrations on such lenses, image warping may, depending on your error criterion, do better than active optics.

± Because image warping, as describe here, is local in nature errors in the localization of calibration features will have a local affect. Such errors, however, will *not* be mitigated, except in spatial extent, by the number of features. If we used a global model for the distortion then we might offset feature localization errors by overconstraining the model.

− For simple (uncorrected) lenses, the active optics approach yields better overall results since it can refocus the channels. We can not quantitatively say how much active optics gains since the image warping results should improve if the images are taken focused with yellow or unfiltered light.

− The active optics approach does need the same amount of calibration information for operation on a number of focus planes. They would only need to compute change in zoom, and shifts for each different depth. The current image warping would need a full cubic-spline mesh for each depth, though a more global model might be developed.

5 Conclusions and future work

This paper demonstrated the idea of image warping for the correction of chromatic aberration using images from two different camera /

*We were not able to determine the importance of this last stage of the CMU active optics approach.

lenses. The method compared reasonably well, both qualitatively and quantitative, to the CMU active optics approach, [Willson and Shafer, 91b]. The proposed warping methods used recently developed image reconstruction/restoration methods [Boult and Wolberg, 91], which were shown to out perform other techniques.

Acknowledgments

This work was supported in part by DARPA Contract #N00039-84-C-0165 and in part by the NSF PYI award #IRI-90-57951 with additional support from Seimans and A&TT. Thanks to Reg Wilson, Steve Shafer and the folks at the CMU Calibrated Imaging Lab for images, the data, the early drafts of the tech. report, and for generally useful discussions about image acquisition.

References

[Boult and Wolberg, 91] T. Boult and G. Wolberg. Local image reconstruction and subpixel restoration algorithms. Technical report, Center for Research in Intelligent Systems, Department of Computer Science, Columbia University, 1991.

[Fant, 86] K.M. Fant. A nonaliasing, real-time spatial transform technique. *IEEE Computer Graphics and Applications*, 6(1):71–80, January 1986. See also "Letters to the Editor" in Vol.6 No.3, pp. 66-67, Mar. 1986 and Vol.6 No.7, pp 3-8 July 1986.

[Green, 89] W.B. Green. *Digital Image Processing: A Systems Approach.* Van Nostrand Reinhold Co., NY, 1989.

[Kingslake, 78] R. Kingslake. *Lens Design Fundamentals.* Academic Press, NYC NY, 1978.

[Klinker, 88] G.J. Klinker. *A Physical Approach to Color Image Understanding.* PhD thesis, Carnegie Mellon University, Pitt. PA, May 1988.

[Krotkov, 87] E.P. Krotkov. *Exploratory visual sensing for determining spatial layour with an agile stereo camera system.* PhD thesis, University of Pennsylvania, Phila. PA, April 1987.

[Laikin, 91] M. Laikin. *Lens Design.* Marcel Dekker, Inc., NY, 1991.

[Park and Schowengerdt, 83] S.K. Park and R.A. Schowengerdt. Image reconstruction by parametric cubic convolution. *Computer Vision, Graphics, and Image Processing*, 23:258–272, 1983.

[Rifman and McKinnon, 74] S.S. Rifman and D.M. McKinnon. Evaluation of digital correction techiques for ERTS images. Technical Report 20634-6003-TU-00, TRW Systems, Redondo Beach, Calif., July 1974.

[Slama *et al.*, 80] C.C Slama, C. Theurer, and S.W. Henriksen. *Manual of Photogrammetry.* American Society of Photogrammetry, Falls Church, VA, Fourth Edition, 1980.

[Willson and Shafer, 91a] Reg Willson and Steven A. Shafer. Active lens control for high precision computer imaging. In *Proceedings of IEEE Conference on Robotics and Automation*, volume 3, pages 2063–2070, Sacramento, CA, 1991.

[Willson and Shafer, 91b] Reg Willson and Steven A. Shafer. Dynamic lens compensation for active color imaging and constant magnification focusing. Technical report, The Robotics Institute, Carnegie Mellon University, 1991.

[Wolberg and Boult, 89] G. Wolberg and Terrance E. Boult. Separable image warping with spatial lookup tables. *Computer Graphics*, 23(3):369–378, July 1989. (SIGGRAPH '89 Proceedings).

[Wolberg and Boult, 91] G. Wolberg and Terrance E. Boult. Imaging consistent reconstruction/restoration and the integrating resampler. Technical report, Center for Research in Intelligent Systems, Department of Computer Science, Columbia University, 1991.

On Finding the Ends of SHGCs in an Edge Image

Hiroaki SATO* **and Thomas O. BINFORD**
Robotics Laboratory
Department of Computer Science
Stanford University, California 94305

Abstract

Algorithms to detect pairs of edges which could be ends of a straight homogeneous generalized cylinder (SHGC) are proposed. Geometrical constraints for the ends of SHGC are utilized to group edgels and edge segments in a complex image. Based on the analysis of computational complexity, it is expected that certain restrictions to object shapes or a priori information are needed to avoid enormous computation. Two methods are investigated. The first algorithm is for a subset of SHGCs where scaling factors of the cross-section at the ends of an SHGC are the same. The second algorithm is for any SHGC. However, a modified version is implemented to reduce computation; given a reference end edge, it finds the edges possibly paired with it. Several examples of ends extracted from real images are reported to show the feasibility and limitation of the algorithms.

1 Introduction

Although shape recovery of three-dimensional forms from a two-dimensional image is underconstrained, it can be simplified when the objects belong to a generic object class. For curved objects, a popular class for object representation is the class of *generalized cylinders* [Binford 71]. A generalized cylinder is the solid obtained by sweeping a planar region, called its *cross-section*, along a space curve, called its *spine* or its *axis* [Agin and Binford 73]. And the class of *straight homogeneous generalized cylinders* (SHGCs) is a subset of generalized cylinders where the solids are obtained by an arbitrary scaling transform of an arbitrary cross-section along a straight axis [Shafer and Kanade 83] [Ponce et al. 89].

Effort has gone into extraction of the axes and limbs. Binford developed an algorithm, called *projection*, and Nevatia and Binford used it to extract the axis and the pairs of the limb from the contours of object boundaries [Nevatia and Binford 77]. Also, algorithms have been developed to find so-called *ribbons*, *smoothed local symmetries*, and *skewed symmetries*, which are two-dimensional projection of some type of generalized cylinders [Brady and Asada 84] [Brooks 81] [Ponce 90] [Sumanaweera et al. 88]. These algorithms are based on the local symmetry of projected edges / limb contours and the continuity of their axes. Ponce et al. clarified invariant properties of SHGCs and developed algorithms to extract the axes of SHGC from their contours [Ponce et al. 89].

In this paper, we report results from algorithms to detect pairs of edges which could be from the ends of an SHGC part. Since ends / cross-sections and limbs / meridians are complementary and essential information to recover the three dimensional shape of an SHGC object, either finding the ends or finding the limbs can be the initial step to generate partial descriptions of objects in an image [Rao and Nevatia 87]. Especially for industrial objects such as pipes or valves, ends are often more apparent and tend to give more useful information than limbs.

It is often difficult to recognize objects in a real image because (1) the image contains noise edges from background, marks, shadows, and specular faces, and (2) the object itself consists of many parts, and the ambiguity in matching a large number of edges causes combinatorial explosion! This work was initiated because little work has been done on the detection of ends or cross-sections despite that strong constraints exist for the ends of SHGCs. Finding the ends of SHGCs raises the level of description from edges to object parts and could make hypothesis generation for objects feasible since it significantly reduces the number of descriptions by applying strong geometric constraints imposed on the ends of SHGCs.

Preliminary results were presented at the IU Workshop September '90. Other researchers show similar results using B-Spline representation of edges in detecting parallel symmetry [Saint-Marc and Medioni 90]. In the recovery of three-dimensional shapes from their two-dimensional projections, it is shown that an SHGC can be recovered with a few degree of freedom when its cross-section and limbs are given. However, most of the previous work show their experiments using computer-synthesized images only [Gross and Boult 90] [Ulupinar and Nevatia 90]. As mentioned in them, detection and computation of symmetries, which could be the limbs and ends of SHGCs, is a difficult task in real images.

The primary problem for finding the ends of SHGCs

*Information Systems Research Center, Canon Inc., 890-12 Kashimada, Saiwai-ku, Kawasaki, Kanagawa 211, Japan. email: sato@cis.canon.co.jp

is computational complexity for testing geometrical constraints. To combat complexity, we introduce restrictions on object shapes or a priori information. The first algorithm is for cylindrical ends SHGCs, a subset of SHGCs where the scaling factors at the ends are the same. The second algorithm is for any SHGC. However, a modified version is implemented to reduce the computation; given a reference end edge, it finds the edges possibly paired with it. (It is extensible to the no a priori information case at the cost of execution time by giving it every edge as a reference edge.) Except for the above restrictions, the algorithms are designed to avoid any implicit limitations to applicable object shapes; they permit arbitrary cross-section shapes, arbitrary angles between the axis and the cross-sections, and arbitrary sweeping rules in the second algorithm. Also noise edges from background, marks, and specular faces may be included in input images. The rest of the paper is organized as follows:

In section 2, we clarify the properties of the ends of an SHGC to make the paper self-contained, though more precise definitions and proofs can be found in the previous work mentioned above. In section 3, we review the previously reported algorithms from the viewpoint of computational complexity. In section 4, we present an algorithm to find the pairs of edge segments which could be from two equally scaled ends of an SHGC. It shows substantial ability to group edges for an SGHC part. In section 5, we propose an algorithm which can detect the ends of any SHGC. Several experimental results of the implemented algorithms for various objects are shown. In this paper, all image data used in the experiments are acquired through TV cameras, and are processed by an edge detector which gives sufficiently good measurement of orientation and position, and links continuous edgels into edge segments.

2 The properties of the ends of straight homogeneous generalized cylinders

In this section, we clarify the properties of the ends of SHGC to make this paper self-contained, though they can be found in some previous work like [Rao and Medioni 88] [Shafer and Kanade 83]. In Ponce et al.'s definition, a straight homogeneous generalized cylinder is the solid swept by a planar cross-section as it is translated and scaled along a straight axis (figure 1). Cross-sections are arbitrary shapes and are not necessarily orthogonal to the axis. The scaling is arbitrary and is governed by a function along the axis, called its sweeping rule. In this paper, we assume the imaging projection is orthographic.

According to [Ponce et al. 89], the contour of an SHGC in the image, when including an oblique SHGC, are given by:

$$\overrightarrow{OQ} = \rho(\theta)r(z)\vec{d} + z\vec{b},$$

$$\begin{aligned}\vec{b} &= \sin\beta_0 \sin(\alpha_0 - \alpha)\vec{w} \\ &\quad +[\cos\beta_0 \sin\beta - \cos\beta \sin\beta_0 \cos(\alpha_0 - \alpha)]\vec{u}, \\ \vec{d} &= \sin(\theta - \alpha)\vec{w} - \cos\beta\cos(\theta - \alpha)\vec{u},\end{aligned}$$

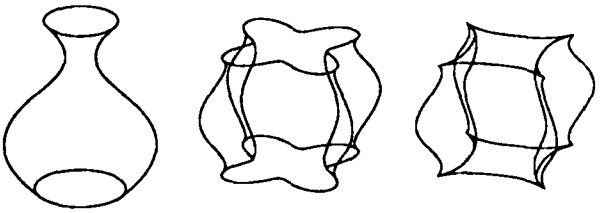


Figure 1: Straight homogeneous generalized cylinders.

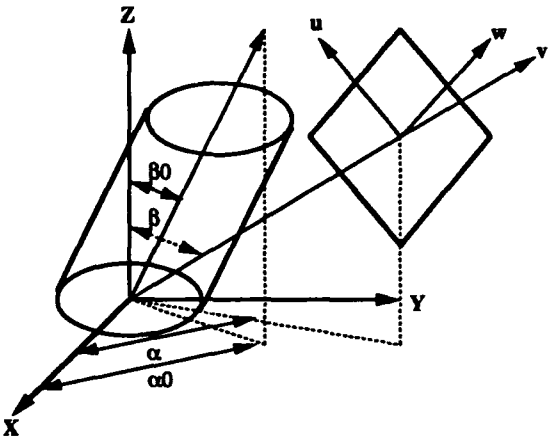


Figure 2: Coordinate system including oblique case.

where $\overrightarrow{OQ}$ is a position in the image; $(\vec{w},\vec{u})$ is unit bases of the imaging plane; $\rho(\theta)$ is a cross-section curve; $r(z)$ is a sweeping rule; (α,β) is a viewing angle; and (α_0,β_0) is orientation of the axis of the SHGC (figure 2). Thus, the parallels or the cross-sections, the contours caused by a constant z value, are observed as follows:

$$\vec{F}_i(\theta) = \alpha_i \cdot \vec{f}_0(\theta) + \vec{t}_i.$$

by letting $\vec{f}_0(\theta)$ be $\rho(\theta)\vec{d}$, α_i be $r(z_i)$, and $\vec{t}_i$ be $z_i\vec{b}$, where i indicates the i-th parallel. In the image, α_i corresponds to the scaling factor of the i-th parallel; $\vec{f}_0$ is the projected cross-section; $\vec{t}_i$ corresponds to the parallel transfer vector of the i-th parallel.

In this paper, we use the term *end pair* or *ends* for the two ends of an SHGC, i.e., the first and last parallel; *co-meridian edgel pair* for a pair of edgels in the ends which lie in a meridian curve; *corresponding co-meridian edgel pairs* for co-meridian edgel pairs which form an end pair; *origin of scaling* for the point at which all the extensions of the lines which pass through each edgel pair in corresponding co-meridian edgel pairs intersect; and *scaling ratio* for the ratio of scaling factors for a co-meridian edgel pair (figure 3).

(I) A pair of edgels which form a co-meridian edgel pair have the same orientation.

In terms of the first derivative of cross-section curves,

$$\begin{aligned}\frac{\partial w}{\partial \theta} &= F'(\theta)\cos\theta - F(\theta)\sin\theta \\ &= \alpha_i(f'(\theta)\cos\theta - f(\theta)\sin\theta),\end{aligned}$$

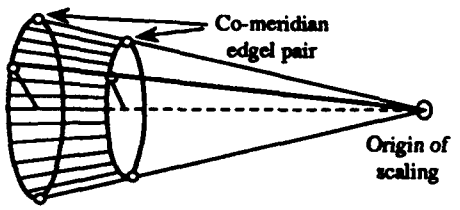


Figure 3: Origin of scaling and co-meridian edgel pair.

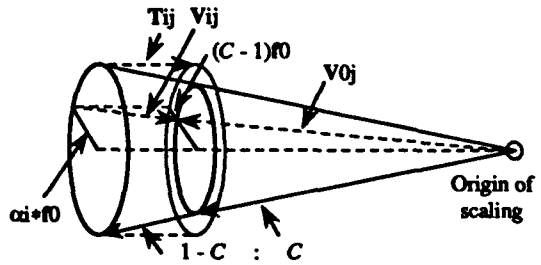


Figure 4: The ends are linear scaling from the origin of scaling.

$$\begin{aligned}\frac{\partial u}{\partial \theta} &= F'(\theta)\sin\theta + F(\theta)\cos\theta \\ &= \alpha_i(f'(\theta)\sin\theta + f(\theta)\cos\theta),\end{aligned}$$

where $F(\theta) = |\vec{F}_i(\theta)|, f(\theta) = |\vec{f}_0(\theta)|$. Thus, orientation of the tangents at the same θ points are the same regardless of their scaling factors.

(II) The extension lines passing through each edgel pair in corresponding co-meridian edgel pairs intersect at a point, their origin of scaling.

Let the points in the extension line be $\vec{G}_{ij}(\theta, l)$. Then,

$$\begin{aligned}\vec{G}_{ij}(\theta,l) &= \frac{\vec{F}_j(\theta) - \vec{F}_i(\theta)}{|\vec{F}_j(\theta) - \vec{F}_i(\theta)|}\cdot l + \vec{F}_j(\theta) \\ &= \left(\frac{\alpha_j - \alpha_i}{|\vec{F}_j(\theta) - \vec{F}_i(\theta)|}l + \alpha_j\right)\vec{f}_0(\theta) \\ &\quad + \frac{\vec{t_j} - \vec{t_i}}{|\vec{F}_j(\theta) - \vec{F}_i(\theta)|}l + \vec{t_j},\end{aligned}$$

where l is the length of extension; α_i, α_j are the scaling factors of i-th and j-th cross-section; $\vec{t_i}$ and $\vec{t_j}$ are the translation vectors of i-th and j-th cross-section. Thus, we get the position of the origin of scaling, $\vec{t_0}$, and the length of the extension to it, l_0 regardless of θ:

$$l_0 = \frac{\alpha_j}{\alpha_i - \alpha_j}|\vec{F}_j(\theta) - \vec{F}_i(\theta)|,$$

$$\vec{t_0} = \vec{G}_{ij}(\theta, l_0) = \frac{\alpha_j}{\alpha_i - \alpha_j}(\vec{t_j} - \vec{t_i}) + \vec{t_j}.$$

(III) The scaling ratio of each edgel pair in corresponding co-meridian edgel pairs is the same (figure 4).

The vector from the origin of scaling to the i-th cross-section $\vec{V}_{0i}(\theta)$, and that to the j-th cross-section $\vec{V}_{0j}(\theta)$ are:

$$\vec{V}_{0i}(\theta) = \vec{F}_i(\theta) - \vec{t_0} = \alpha_i(\vec{f}_0(\theta) + \frac{\vec{t_j} - \vec{t_i}}{\alpha_j - \alpha_i}),$$

$$\vec{V}_{0j}(\theta) = \vec{F}_j(\theta) - \vec{t_0} = \alpha_j(\vec{f}_0(\theta) + \frac{\vec{t_j} - \vec{t_i}}{\alpha_j - \alpha_i}).$$

Thus, the cross-sections are linear scaling of the original cross-section from the origin of scaling. And the ratios of the corresponding lengths are the same at any θ.

(IV) The vector between each edgel pair in corresponding co-meridian edgel pairs is the same when its scaling ratio is equal to 1.

A vector between a co-meridian edgel pair, $\vec{V}_{ij}(\theta)$, is calculated as:

$$\begin{aligned}\vec{V}_{ij}(\theta) &= \vec{V}_{0j}(\theta) - \vec{V}_{0i}(\theta) \\ &= (C-1)\vec{f}_0(\theta) + \vec{T}_{ij}.\end{aligned}$$

where $C = \alpha_j/\alpha_i; \vec{T}_{ij} = (\vec{t_j} - \vec{t_i})$ is the translation vector between i-th and j-th cross-sections. Thus, the vectors are the same regardless of θ when the value of C is 1.

(V) The curvature at a point in an end is proportional to the reciprocal of its scaling factor.

Curvature in an end, κ, is calculated as:

$$\begin{aligned}\kappa &= \frac{2F'(\theta)^2 + F(\theta)^2 - F''(\theta)F(\theta)}{(F'(\theta)^2 + F(\theta)^2)^{3/2}} \\ &= \frac{2f'(\theta)^2 + f(\theta)^2 - f''(\theta)f(\theta)}{\alpha(f'(\theta)^2 + f(\theta)^2)^{3/2}}.\end{aligned}$$

Thus, the curvature is proportional to the reciprocal of the scaling factor α. By using the scaling ratio $C = \alpha_j/\alpha_i = \kappa_i/\kappa_j$ and the positions of a corresponding pair, the position of origin of scaling can be calculated.

3 Computational complexity of finding symmetries

In this section, we overview the previous work for finding symmetries. Though they analyze the limbs of generalized cylinders, the algorithms used there have similarity to the ones shown in section 4 and 5 because these type of algorithms find pairs of edgel which satisfy certain geometrical constraints. We don't refer to the precise geometrical constraints here since they are different for each type of symmetry, but consider the computational complexity of the algorithms.

First, the difficulty of the problem can be classified into the cases where the verification of the constraints requires only local measurements, such as position, orientation, and curvature, of a pair of edgel, or not. In the former case, the simplest implementation of the algorithm has two steps; generate every pair of edgel, and then verify the geometrical constraints for each pair of edgel. This algorithm's complexity is $O(n^2)$ where n is the number of edgels, as shown in Brady's algorithm for finding smoothed local symmetries [Brady and Asada 84]. (They also showed another algorithm which fits curves to the edges and finds symmetries analytically.)

Secondly, the computational complexity depends on the complexity of input data; a simple curve / edge,

such as a boundary of an object, or multiple curves / edges. In the former case, the complexity of the algorithm can be reduced to $O(nk)$ by using projection technique [Nevatia and Binford 77] (where k is the number of discretized orientations). A rough description of the algorithm is: discretize the possible directions of the correspondence between an edgel pair; for each of these directions, project all edgels into buckets, and verity the conditions for edgel pairs within the same bucket; lastly, group the resulting pairs into symmetries. Ponce also used this method for finding skewed symmetries [Ponce 90].

When multiple edges are given as input data, such as an edge image extracted from an ordinary intensity image, still simple implementation can be applicable in the complexity of $O(n^2)$. This can be reduced to $O(nke)$ by using projection, where e is the number of edges, assuming that only one edgel is put to each bucket for each edge, as mentioned in [Sumanaweera et al. 88].

When it is not possible to determine whether a pair of edgels satisfy the constraints with their local measurements only, in other words, when the constraints include one or more free variables to be selected globally, a hypothesis generation and verification process like Hough transform [Hough 62] is required. Then its complexity becomes, for instance, $O(n^2d)$ in the one free variable case, where d is the number of hypotheses for a pair. Ponce et al. used this type of algorithm to find axes of SHGCs from their limb contours [Ponce et al. 89]. (They also implemented another algorithm which use curvature feature points instead of all the edgels.)

In the above discussion, we concentrated on the grouping of the edgel pairs and neglected necessary post-processings. Most of the algorithms require post-processings, such as selection of a group in hypotheses, restoration of the pairs of edge segments from the pairs of edgels. Some of the algorithms in the previous work, such as [Brady and Asada 84] [Ponce et al. 89] and [Saint-Marc and Medioni 90], use fitting of curves to edges or selection of edgel points by certain geometrical features. We didn't examine their computational complexity, since these pre-processings themselves are not simple and their complexities don't depend on the number of edgels.

It is natural that the required computational complexity depends on both the geometrical constraints between the edge pair to be extracted and the number of possible edge pairs in input data. Though it is difficult in general to estimate computing time for an algorithm, we can guess it for the silimar types of algorithms which verify the constraints on edgels. In [Ponce 90], the reported computing time by an $O(nk)$ algorithm is around 30 seconds. Since the number of edgels n increases by the number of edges e, $O(nke)$ algorithms for multiple edges should take much longer time. The reported computing time by an $O(nke)$ algorithm in [Sumanaweera et al. 88] is about 1.5 hour. And in the previous work, the authors presented alternate algorithms for the $O(n^2)$ algorithms. According to these facts and our experiences in implementing the algorithms proposed in this paper, it is expected that for finding edgel pairs by verifying the geometrical constraints, the algorithms which require more than $O(n^2)$ computation are often impractical for the computational power of current sequential computers, and that $O(nke)$ algorithms tend to take a fairly long time. Since accurate measurement of the curveture values may not be available in an edge image, the computational complexity of finding the ends of SHGCs is roughly $O(nkdc)$, where c is the number of partially parallel edges, and d is the number of hypotheses for a pair of edgels. (In this paper, we assume that a lot of edges are given as inputs to the algorithms and the constraints for the ends of SHGCs leave one free variable without curvature measurement. See section 4 and 5.) This suggensts that certain restrictions may be required to avoid enormous execution time. In the following sections, we investigate two possibilities; one is the restriction of the shapes, the other is a given a priori information.

4 Simplified algorithms in the case of cylindrical ends

The algorithm to find the ends of SHGCs can be simplified significantly for cylindrical ends, which means the scaling factors at the both ends are the same. For example, the center and right objects in figure 1 have cilindrical ends. Note that 'cylindrical' doesn't mean circular cross-sections. For cylindrical ends, the origin of scalings are infinitely distant, and the scaling ratio is 1. Every pair of edgels that has the same origin of scaling makes the same projection angle. Thus, we have more strict constraints for the ends of cylindrical SHGCs as follows:

(i) A pair of edgels which form a co-meridian edgel pair have the same orientation.

(ii) The vectors between each edgel pair in corresponding co-meridian edgel pairs have the same orientation.

(iii) The vectors between each edgel pair in corresponding co-meridian edgel pairs have the same length.

By using projection, we can get the algorithm to detect cylindrical ends, algorithm 1, summarized as follows:

Algorithm 1

1. discretize orientation for projection direction, prepare buckets for projection,
2. group edgels according to their orientation into orientation-groups,
3. for each projection direction,
   (a) for each orientation-group, project edgels in the group into buckets; (The result is projection-orientation-groups.)
   (b) generate all pairs of edgels in each projection-orientation-group;
   (c) calculate the distance between each edgel pair, and group the pairs according to their distances into distance-groups,
   (d) for each distance-group, restore the edge-pairs which correspond to the edgel-pairs in the group.

The computational complexity of this algorithm is roughly $O(nkc)$, where c is the number of the edgels which have the same orientation in a bucket, i.e., the number of partially parallel edges. More precisely, the grouping by the above constraint (i), orientation of an edgel, requires $O(n)$ work because it requires the local measurement of each edgel only. The grouping by the constraint (ii) and (iii) require $O(nkc)$ because the average number of edgels in a projection-orientation-group is c and that is the average number of edgel pairs per edgel in a projection direction. Edgel-pairs in each group are back-projected to edges and edge-pairs are selected if a large part of the original edgels are recovered.

Examples of the input image to the implemented program, which is the edges detected in an intensity image, and the results of the program are shown in figure 5 and figure 6. (In these images, some edges from background are removed but it is not necessary as shown in figure 9.) It takes about 15 minutes to group the possible end-edgel-pairs. In the figures, sets of parallel edge segments are shown as examples of the program's results. In figure 5(b), the algorithm grouped three edge segments and recovered most of the ends for a cylinder part, though they have considerable gaps among them. In figure 5(c), the algorithm detects many parallel curves. Although we don't have techniques to group such repetitive structure, a clue for that type of structure can be acquired. In figure 5(d), the algorithm detects parallel straight lines. Though they are neither ends nor cross-sections, the algorithm should detect parallel straight lines to cope with polyhedral objects. Simple curves like ellipses and straight lines are shown in figure 5, however, there are no restriction on the shape of cross-section. In the figures, we find that the algorithm has the ability to group the edges into the sets of parallel edges, so that the results of the algorithm could be used to generate hypotheses of SHGC parts for recognition.

5 Finding the ends of SHGCs

In this section, we propose an algorithm to detect the ends of any SHGC. However, to avoid enormous computation, the algorithm is modified to cope with a given reference end edge. It is extensible to no a priori information case simply by giving it every edge as a reference end edge. Also, it is easy to predict the performance of the algorithm in the case with no a priori information because there is no difference in the input data or grouping functions and the implemented algorithm results a subset of all of the results. The constraints for the ends of SHGC are as follows:

(i) A pair of edgels which form a co-meridian edgel pair have the same orientation.

(ii) The extensions of the lines passing through each edgel pair in corresponding co-meridian edgel pairs intersect at an origin of scaling.

(iii) The scaling ratio of each edgel pair in corresponding co-meridian edgel pairs is the same.

(iv) When the scaling ratio is nearly equal to 1, the distance between each edgel pair in corresponding co-meridian edgel pairs is nearly the same.

In order to deal with cylindrical ends, the fourth constraint are used. However, to make the explanation simple we don't refer to the fourth constraint in this section. (See section 4 for the fourth constraint for the ends.)

Availability of curvature measurement effects the computational complexity of algorithm to detect the ends of SHGCs because the position of the origin of scaling and the scaling-ratio can be calculated from the positions and curvatures of a co-meridian edgel pair. If we could measure curvature accurately, the simplest algorithm to detect the ends would be summarized as follows: discretize the position in two-dimensional space, and prepare cells for origin of scaling; for each pair of edgels, verify the constraint of their orientation, calculate the scaling ratio C_i and position of origin of scaling (r_i, ϕ_i), then put the edgel-pair into the discretized cell indexed by (r_i, ϕ_i, C_i); get the group of edgel-pairs in each cell.

The complexity of this algorithm is $O(n^2)$ except for the post processings to determine the edge-pairs which correspond to the edgel-pairs. This can be reduced to $O(nkc)$ by using projection where n is the number of edgels, k is the number of discretized directions, and c is the number of partially parallel edges.

The difficulties are caused by the facts, (a) the current measurement of curvature is not accurate enough to classify the edgel-pairs with their ratio nor to determine the position of origin of scaling, (b) the position of an origin of scaling may be infinitely distant, so that it may require infinite number of cells without non-linear discretization.

To solve the former difficulty, the algorithm shown below hypothesizes origin of scalings in the direction of projection. Same as a Hough transform, this method requires post processing to determine the position of origin of scaling.

For the latter difficulty, we select a polar coordinate system (r, ϕ) and use non-linear scaling function to roughly normalize the error. It is selected because the error in the angle is roughly independent of the length of the extension, except around the origin of coordinates, and the error in radius is magnified proportionally to the length of extension, i.e., the distance between an edgel to the origin of scaling. A scaling function $a/(r+b)$, where a and b are constants, is selected because it can be approximated to a/r, a reciprocal function when $r \gg b$, and approximated to $\frac{a}{b} - \frac{a}{b^2}r$, a linear function when $r \ll b$ (figure 7). Here we propose an alternate algorithm summarized as follows (see figure 8 also):

Algorithm 2

1. discretize orientation for projection direction, discretize two-dimensional space for origin of scaling, and prepare buckets for projection and cells for origin of scaling,
2. group the edgels according to their orientation, (the results are called orientation-groups,)
3. for each projection direction, and for each orientation-group,
   (a) project edgels in the group into buckets, (the results are called projection-orientation-groups,)

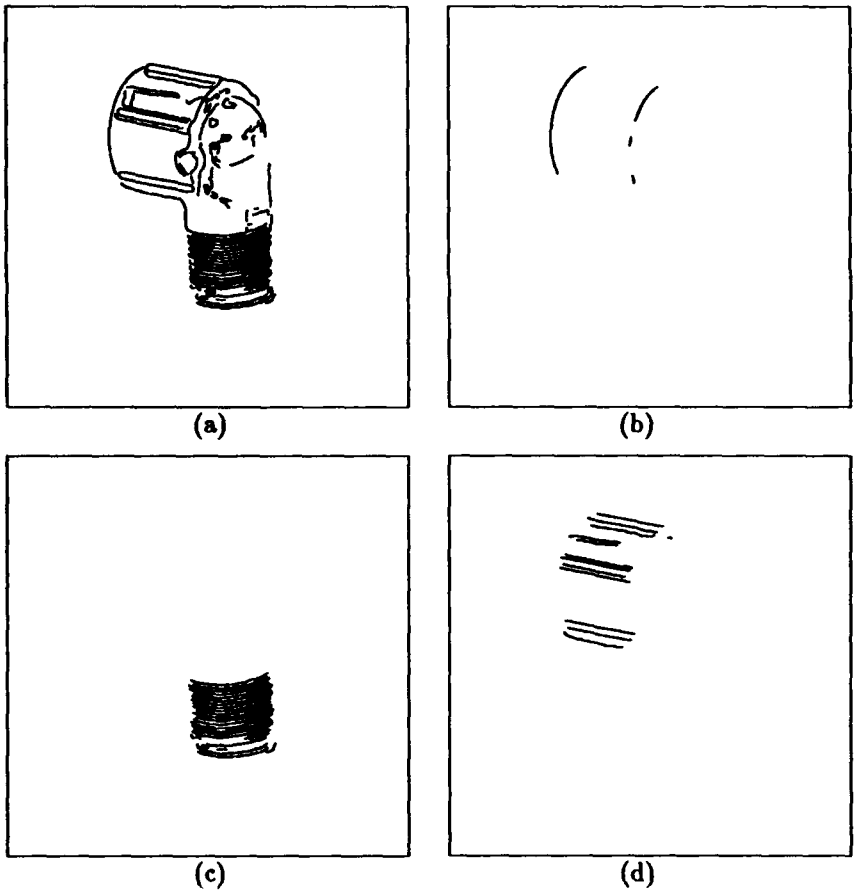


Figure 5: Examples of the extracted parallels by algorithm 1.

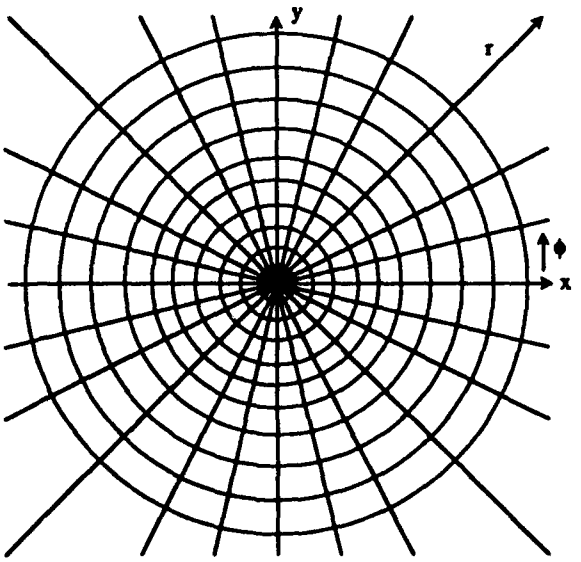


Figure 7: Accumulater cells for origin of scalings in algorithm 2.

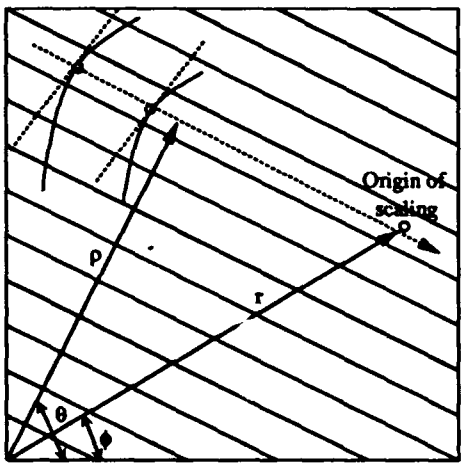


Figure 8: Projection and position of origin of scaling.

(b) calculate the extension lines in the projection direction and determine the cells (r, ϕ) which each extension line passes through,

(c) put the projection-orientation-group into the cells,

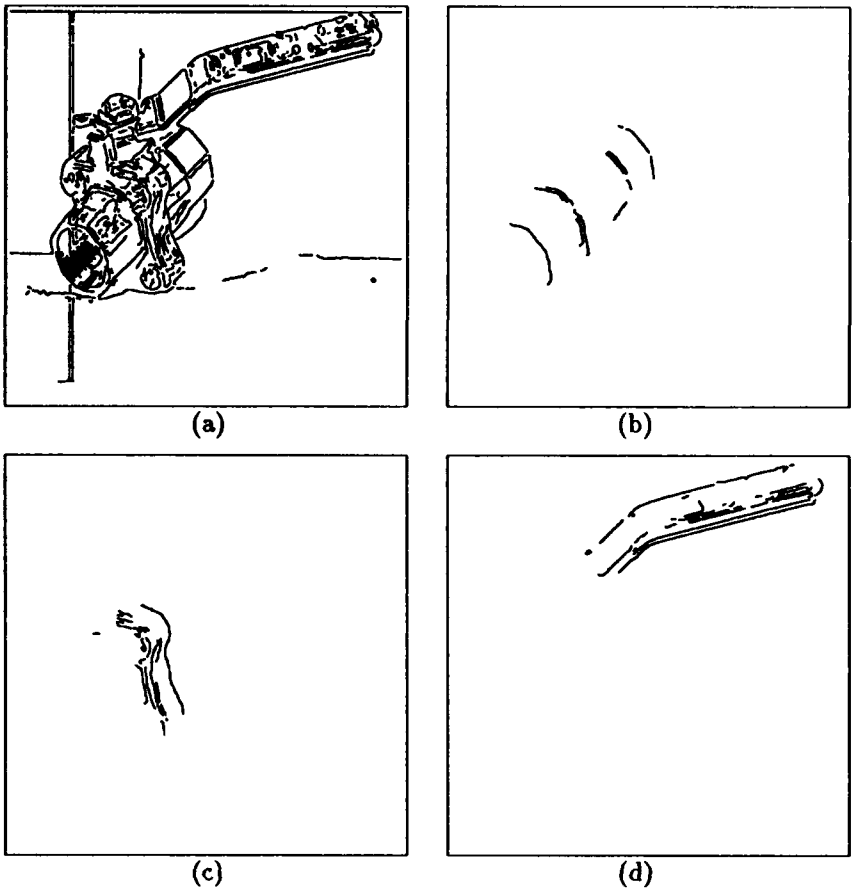


Figure 6: Other examples of the extracted parallels by algorithm 1.

4. for each cell,
   (a) calculate the scaling ratio for each edgel pair in each projection-orientation-group in the cell,
   (b) group the edgel-pairs according to their scaling ratios and position of the cell, (the results are called scaling-origin-groups,)
5. for each scaling-origin-group, restore the edge-pairs corresponding to the edgel-pairs in the group.

The computational complexity of the above algorithm is roughly $O(nkdc)$, where d is the number of hypothesized origins of scaling for an edgel-pair. Unfortunately, this algorithm still takes a very long time. For instance, it would take a few days on Symbolics 3600 to get the results from the data shown in figure 9, so that it is not practical even for the laboratory use. We reduce it by giving a reference edge as an end of an SHGC and pre-select the candidate positions of origin of scaling. The above algorithm is modified and the above step 4 is changed to the followings:

4-I select candidates for the position of origin of scaling,

4-II for each candidate cell (r, ϕ)
   (a) for each projection-orientation-group in the cell, generate pairs of edgel in the group which consist of an edgel in the reference edge and another in the other edges, then calculate the scaling ratio for each pair,
   (b) group the edgel-pairs according to their scaling-ratios and position of the cell into scaling-origin-groups,

Hereafter we call this modified algorithm algorithm 2'. The precise description of the pre-selection of the candidates in step 4-I is presented in the appendix A. The complexity of this algorithm is much less than $O(nkdc)$; step 2 requires $O(n)$, step 3 requires $O(nkd)$ at most, step 4-I requires $O(nkd)$, and step 4-II requires $O(mpc)$ where m is the number of edgels in the reference edge and p is the number of candidates for origin of scaling. Also, the selection of the edge-pairs, step 5, which is another time-consuming process, needs execution time roughly proportional to the number of edgel-pairs, mpc.

Figure 9 and 10 shows examples of the results by algorithm 2'. The input edges are supposed to be divided at the peak of curvature. In figure 9, most of the end edges are visible, so that detection of the ends is quite stable even among the noise edges from background, marks, or specular faces. In figure 10, it detects many false positives for a straight edge because every parallel straight edges satisfy the constraints used in the algorithm, however, for curved edge, it results few false positives. The program takes about an hour with 30 candidates for ori-

gin of scaling. Because of discretizing error and ranges for grouping, it may find false positives. However, the experimental results show only a small number of false positives except for parallel straight lines, since the geometrical constraints derived from the definition of SHGC are sufficiently strict to reject most false correspondences when collected along an edge.

6 Conclusion

In this paper, we proposed two algorithms to detect the pairs of edge segments that could be from the ends of an SHGC by using geometrical constraints derived from the shape restriction of SHGCs. Algorithm 1 is for the subset of SHGCs where scalings at the both ends are the same. This restriction to the object shapes gives strict constraints, and reduces the computational complexity of the algorithm. Algorithm 2 is for any SHGC and is supposed to detect all the pairs of edges which could be from the ends of an SHGC. However, to avoid an enormous computation time, a modified version which copes with a given reference end edge is implemented. Both algorithms are implemented on Symbolics 3600 and show their performances to detect the ends of SHGCs in a reasonable time.

The algorithms employ parameters which determine the range of feature values for a group. They depend on the error in local measurements, position and orientation, and the deformation by perspective projection in real images. If the deformation were visible, then a part of the algorithms should be changed. Under perspective projection, the edgels that form a co-meridian edgel pair do not have the same orientation, but their tangent lines intersect in a line. (See appendix B for its proof.) Thus, the second step of the algorithms is replaced with the following: project each edgel to a bucket, generate edgel-pairs in a bucket, calculate intersection of tangent lines, use Hough transform to group the edgel-pairs. This algorithm requires $O(n^2d)$ or $O(nkde)$ computation. Unfortunately, this algorithm may not be practical without some restrictions and is not implemented. Instead, we use loose parameters in the experiments. The loose parameters absorb the measurement error and the deformation to some extent, so that the algorithms tend to detect all the end pairs and false positives. Even with fairly loose parameters, the detected false positives are not so many in experiments. This fact shows that even though the geometrical constraints at a point is weak, those constraints collected along an edge become strong enough to exclude most of false positives.

Accurate measurement in the edge detection may significantly reduce the computation time required to find the ends of SHGCs. If we got reliable curvature values, the computational complexity to find the pairs of the corresponding edges would be $O(nkc)$ and detection of the ends of any SHGC could be completed in a reasonable time. Though it would be reduced further by using curvature feature points or curve fitting [Saint-Marc and Medioni 90], reliable and efficient pre-processings may still be another difficult problem.

Finding the limbs or meridian edges in real images becomes more reliable and efficient when we have the information from the end pairs. And as shown in previous work, it is possible to recover 3D shape of SHGCs when we get the cross-sections and the meridians or limbs, with other information, such as shading or an assumption of skewed symmetries. So, recovering 3D shape of SHGCs from the edges in real images is natural extension of this work [Sato and Binford 92].

The segmentation or grouping ability of the algorithms are especially useful for hypothesis generation in the model-based image recognition systems. Since the geometrical constraints for the ends of an SHGC exclude most of the false edge pairs, the algorithms detect only a limited number of end pairs for the SHGC parts in an object. In the context of the SUCCESSOR model-based vision system, the detected ends of SHGCs, combined with other visual clues such as skewed symmetry and ribbons, would be used to hypothesize the partial faces and volumes, which are further matched to the models represented in the whole-part graph of SHGCs.

Acknowledgement The authors thank Wally Mann for his contribution for the early draft and various discussions throughout this work.

A Pre-selection of candidate positions for origin of scaling

Though detection of the ends of an SHGC requires grouping the edgel-pairs by their properties, we reduce the computation for them by selecting candidate positions for origin of scalings. The pre-selection can be done by applying a looser necessary condition, i.e., an edgel in the reference end edge should have parallel edgels in the same projection bucket. Thus, the following method is used to select the candidates.

1. for each cell for origin of scaling,
    2. for each projection-orientation-group,
        3. accumulate the number of the groups containing edgels in the reference edge, Ne,
        4. accumulate the number of the groups containing both edgels in the reference edge and edgels in the other edges, Nc,
    5. calculate the ratio of Nc to Ne,
6. sort the ratio and return the cell positions which give high scores.

B A constraint for the ends of SHGC in perspective projection

Constraint: Under perspective projection, for a pair of points in ends which lie in an meridian, the tangents to the contours at these points intersect in a line if they intersect, otherwise they are parallel to the line.

Proof: Tangents to the contours at these points are parallel in 3D and are on a plane made by the line between the points and the tangent line. Parallel lines in 3D space intersect at a point in perspective projection unless they are parallel in the imaging plane. Suppose a 3D plane which contains the camera position and is

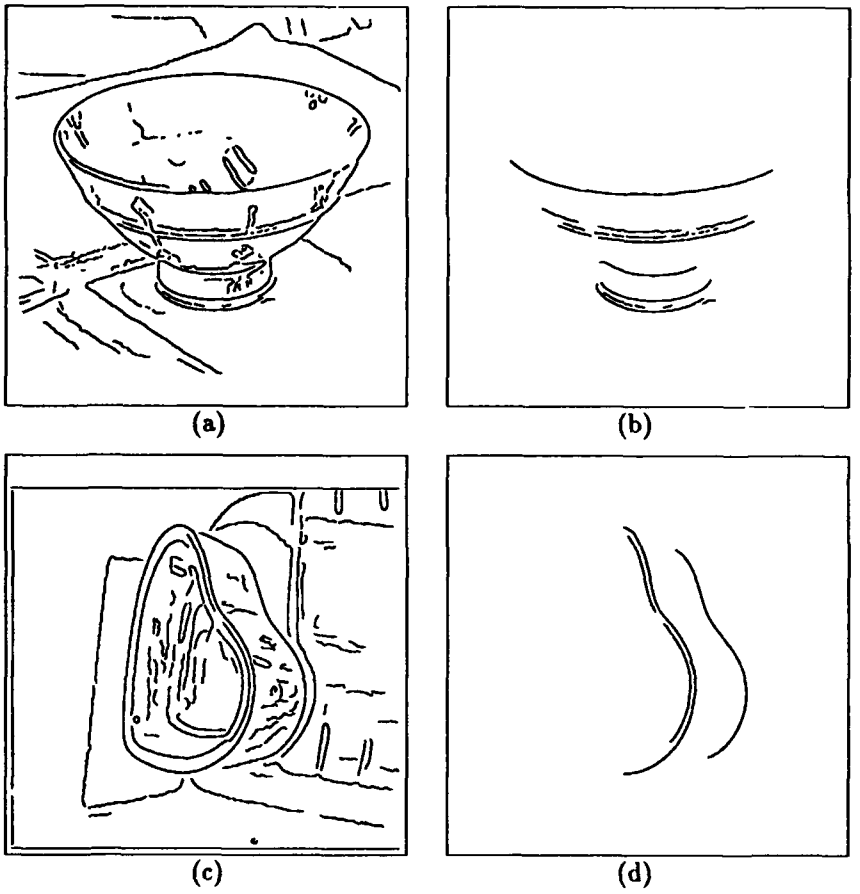


Figure 9: Examples of the extracted ends of SHGC by algorithm 2'.

parallel to the plane including an end face, the intersection of the plane and extension of the side face makes a virtual cross-section which is projected to a line in imaging plane. The tangent lines at the corresponding points in a meridian, a pair of in the ends and a point in the virtual cross-section, intersect at a point in the imaging plane and the intersection is in the line caused by the projection of the virtual cross-section, unless they are parallel in the image plane.

References

[Agin and Binford 73] Agin, G.J., and Binford, T.O., "Computer description of curved objects," Proc. Third IJCAI, pp.629-635, Aug. 1973.

[Binford 71] Binford, T.O., "Visual perception by computer", Proc. IEEE Conf. on Systems and Control, Miami, December, 1971.

[Brady and Asada 84] Brady, J.M. and Asada, H., "Smooth local symmetries and their implementation," International Journal of Robotics Research, vol.3, no.3, 1984, pp.36-60.

[Brooks 81] Brooks, R.A., "Symbolic reasoning among 3D models and 2D images," Artificial Intelligence 17, pp. 285-348, 1981.

[Gross and Boult 90] Gross, A. D. and Boult, T. E., "Recovery of generalized cylinders from a single intensity view," Proc. of the Image Understanding Workshop, 1990, pp.557-564.

[Hough 62] Hough, P.V.C., "Method and means for recognizing complex patterns," U.S. Patent 3,069,654, 1962.

[Nevatia and Binford 77] Nevatia, R. and Binford, T.O., "Description and recognition of complex curved objects," Artificial Intelligence 8, pp. 77-98, 1977.

[Ponce 90] Ponce, J., "On characterizing ribbons and finding skewed symmetries," Computer Vision, Graphics, and Image Processing, vol.52, no.3, pp.328-340, 1990.

[Ponce et al. 89] Ponce, J., Chelberg, D., and Mann, W., "Invariant properties of straight homogeneous generalized cylinders and their contours," IEEE trans. on PAMI, vol.11, no.9, pp.951-966, 1989.

[Rao and Medioni 88] Rao, K. and Medioni, G., "Useful geometric properties of the generalized cone," Proc. of Computer Vision and Pattern Recognition, 1988, pp.276-281.

[Rao and Nevatia 87] Rao, K. and Nevatia, R., "From sparse 3-D data directly to volumetric shape de-

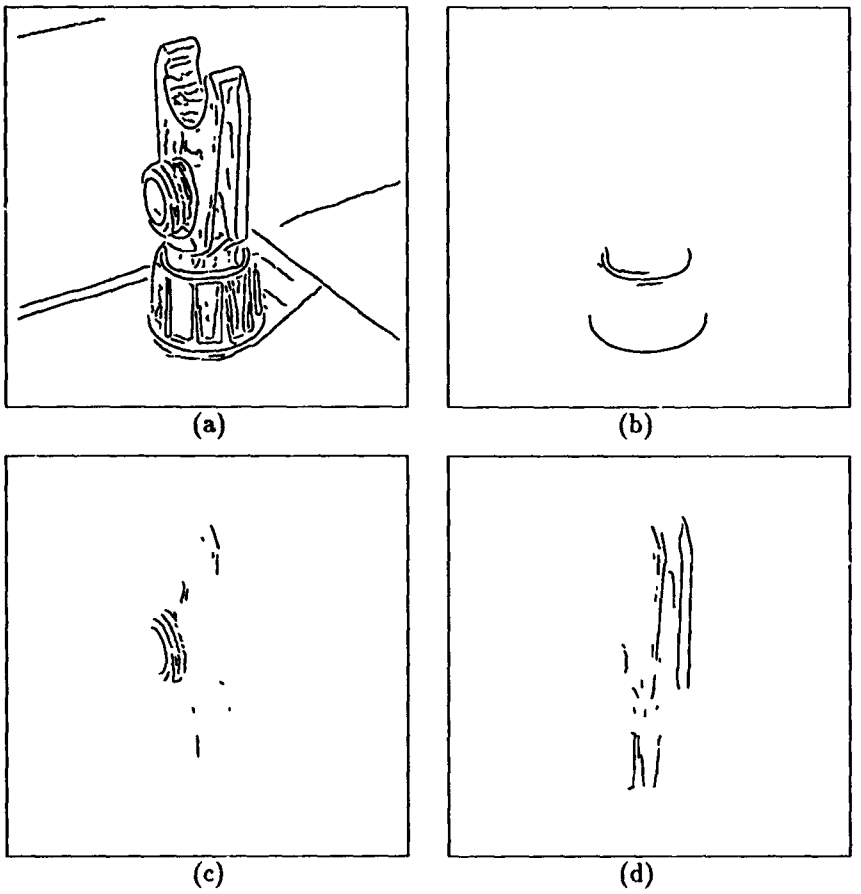


Figure 10: Other examples of the extracted ends of SHGC by algorithm 2'.

scription," Proc. of the Image Understanding Workshop, 1987, pp.360-369.

[Saint-Marc and Medioni 90] Saint-Marc, P. and Medioni, G., "B-spline coutour representation and symmetry detection," Techinical Report IRIS262, University of Southern California, 1990.

[Sato and Binford 92] Sato, H. and Binford, T.O., "BUILDER-I: a system for the extraction of SHGC objects in an edge image," Proc. of the Image Understanding Workshop, 1992.

[Shafer and Kanade 83] Shafer., S. A. and Kanade., T., "The theory of straight homogeneous generalized cylinders," Technical Report CS-083-105, Carnegie-Mellon University, 1983.

[Sumanaweera et al. 88] Sumanaweera, T., Healey, G., Lee, B., Binford, T.O., and Ponce, J., "Image segmentation using geometric and physical constraints," Proc. of the Image Understanding Workshop, 1988, pp.1091-1099.

[Ulupinar and Nevatia 90] Ulupinar, F. and Nevatia, R., "Recovering shape from contour for SHGCs and CGCs," Proc. of the Image Understanding Workshop, 1990, 544-556.

Subsection B:

Stereo

Stereo Using The Displacement Representation

Steve White *
Massachusetts Institute of Technology
Artificial Intelligence Laboratory
545 Technology Square, Cambridge, Massachusetts 02139

Abstract

An important aspect of successful processing of image data into representations for higher level analysis involves preserving such feature parameters as position, location, contrast and blur, while substantially compressing the data into a sparse or compact form. In spite of this, most "edge finders" eliminate this information almost immediately in the process of converting the continuous image into a binary edge map. An alternative representation concept is proposed which preserves these measures through a simple nonlinear transform. It should be applicable to numerous vision problem domains such as stereo, motion and recognition. This representation, called the Displacement Representation, can then be utilized using simple algorithms to extract higher level representations such as stereo disparity, optical flow, and possibly others. A stereo algorithm is described and tested which demonstrates the utility of the approach as well as the integrity of the resulting disparity measures.

1 Introduction

The goals of early vision algorithms are usually stated as *a)* to process image data in such a way as to produce new representations which are rich in feature related information useful for higher level tasks such as stereo depth, motion, and object recognition, and *b)* to substantially reduce signal bandwidth. Thus, if successful, the resulting representation should be sparse without significant loss of feature content, while noise and spurious features would be, by and large, eliminated.

In usual practice, however, the extraction of symbolic representations at the earliest stages obviates this goal. Many edge finders, for instance, take intensity steps in scenes as features using a variety of methods [1, 2, 3, 4, 5]. The resulting representation is usually a bit map where a pixel is marked if, by using some criteria, it is deemed to be near some image feature. By using this encoding, pixel level acuity is imposed on such parameters as location and orientation while others, such as contrast and focus, are simply lost. Subsequent operations such as grouping or filtering cannot be expected to reverse this. At the same time, these algorithms are often prone to marking noise as readily as edges, given the small support of the operators [9].

An alternative approach is found in linear correlation models which preserve the continuous image representation while using various linear operators to attack problems such as stereo [8] and recognition [10]. These methods, despite some significant drawbacks, have often produced some promising results. Ultimately, however, symbolic reasoning *must* be involved for the system to be of any utility in a recognition scheme. Less progress has been made on this approach, in general, than the strongly symbolic approach of edge extraction and analysis. This is perhaps due to the fact that these methods tend to be intolerant of such common imaging characteristics as varying scene illumination and foreshortening with rotation.

An alternative approach is described here, whereby a simple nonlinear transform on the image produces a representation which can be used in higher level vision domains. The properties of edge features such as position, orientation, contrast and blur are preserved in a sparse representation with low noise characteristics. I call it the Displacement representation. It appears applicable in a number of problem domains such as motion, stereo, and object recognition. This paper will develop the stereo model and show some results on real image pairs.

2 The Displacement Representation

The Displacement representation is intended to be a continuous representation which, at any point in the image, compactly encodes 1) the distance to the nearest contrast edge, 2) the orientation of that edge, 3) the contrast of the edge, and 4) the sharpness of the focus. Should such a representation be computable, such things as optical flow and stereo disparity could be obtained by such simple linear operations on the Displacement function as

*This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by an Office of Naval Research University Research Initiative grant under contract N00014-86-K-0685, and in part by the Advanced Projects Agency of the Department of Defense under Army contract number DACA76-85-C-0010 and under Office of Naval Research contract N00014-85-K-0124.

temporal differentiation or spatial subtraction. The Displacement "function", as used here, is defined as that part of the representation which encodes the feature distance.

For example, taking the 1D image $I(x)$ of Figure 1, assume that an image intensity step exists at some location x_o in the image domain. The Displacement function $d(x)$ would encode the distance of any point to x_o, i.e. it would be proportional to $x - x_o$. The representation would also encode contrast, blur and orientation, but these are not shown here.

If you had two such inputs, say one from the left eye and one from the right, and you subtracted their respective Displacement functions, the resulting function would represent the stereo disparity in the neighborhood of the features (see Figure 2). If you took the temporal derivative of the Displacement function, $\dot{d}(x)$, the motion of the image features, or optical flow, could be measured. For this reason, finite differences or differentiations such as this of Displacement functions are called Disparity functions.

The 2D Displacement representation will take as its input the grey-scale image which has been convolved with the Laplacian of a Gaussian. This is a commonly used model of the retinal center-surround response in biological vision. Edges are defined as image events modelable as isolated piecewise linear image boundaries discontinuities in intensity. The degree to which edges must be approximated as linear or isolated depends on the width of the convolving Gaussian.

In the following section, I will derive a version of the Displacement representation appropriate for 1D image slices. Orientation will not be encoded by this simplified approach. Instead of the Laplacian of Gaussian convolved input, a simple second derivative of a 1D Gaussian is used.

The general 2D version of this model is developed next. This is then applied to the problem of stereo correspondence using a scale space scheme.

2.1 The 1D Displacement Model

The Displacement model is predicated on image functions being composed of superposed "edges", or step intensity features:

$$d_o(x) = x - x_o$$
$$I(x) = \alpha u(d_o(x))$$

where $u(x)$ is the unit step function, $d_o(x)$ is the distance from any point x to the edge position x_o, and α establishes both the sign and contrast of the edge feature. Obviously, any arbitrary discrete image can be described using such features, but for the purposes of this model, at some scale features are assumed to be isolated. This issue of what constitutes "isolated" and what happens when edges aren't isolated will not be dealt with in any greater depth here.

The processing starts with the image convolved with a Gaussian of some width σ_b, and the second derivative taken:

$$\begin{aligned} g(x) &= \frac{1}{\sqrt{2\pi}\sigma_b} e^{\frac{-x^2}{2\sigma_b^2}} \\ I''(x) &= g''(x) * I(x) \\ &= \alpha g'(d_o(x)) \\ &= -\alpha \frac{d_o(x)}{\sigma_b^2} g(d_o(x)) \end{aligned}$$

This is a 1D approximation to the retinal operation on the image. A second representation, $I'(x)$, is calculated, either through numerical integration of $I''(x)$, convolution of the image with a $g'(x)$ operator, or simply taking the derivative of the Gaussian smoothed image. In any case, the resulting function for the isolated image feature $I(x)$ is:

$$\begin{aligned} I'(x) &= \int_{-\infty}^{x} I''(x)dx \\ &= \alpha g(d_o(x)) \end{aligned} \quad (1)$$

The Displacement function is simply the ratio of these two representations:

$$\begin{aligned} d(x) &= -\sigma_b^2 \frac{I''(x)}{I'(x)} \\ &= d_o(x) \end{aligned}$$

In other words, by taking the ratio of these two image functions, we can arrive at an approximation to the distance function d_o. This function d is the 1D Displacement representation. The denominator, $I'(x)$, which encodes contrast, is discussed in the next section.

2.2 1D Weighting Function

One issue is determining the reliability of the $d(x)$ estimate when the image is corrupted by additive normal noise. By inspection, the denominator of $d(x)$ is Gaussian (see equation 1) and, as such, will approach zero as $|d_o(x)|$ increases. When $|x - x_o|$ is large, then, we would expect $d(x)$ to be a less reliable estimate of feature proximity than when x is close to the edge. The denominator magnitude is an intuitively reasonable estimate of the merit, or weight, of the $d(x)$ estimate. A recent study has demonstrated the validity of this weighting function.

If one poses the problem of finding the optimal estimation of x_o in the best least squares sense using the Gaussian denominator $I'(x)$, it turns out that the solution is the weighted average of $d(x)$ where the weight is the denominator $I'(x)$ scaled by $g(x - x_o)$:

$$w(x) = I'(x)g(x - x_o) \quad (2)$$

$$x_o = \frac{\int w(x)\,[x - d(x)]\,dx}{\int w(x)dx}. \quad (3)$$

One way of effecting the $g(x - x_o)$ function is by convolving the displacement and weighting functions with the Gaussian $g(x)$. Although convolution introduces a speed penalty — turning a process which is $O(1)$ per

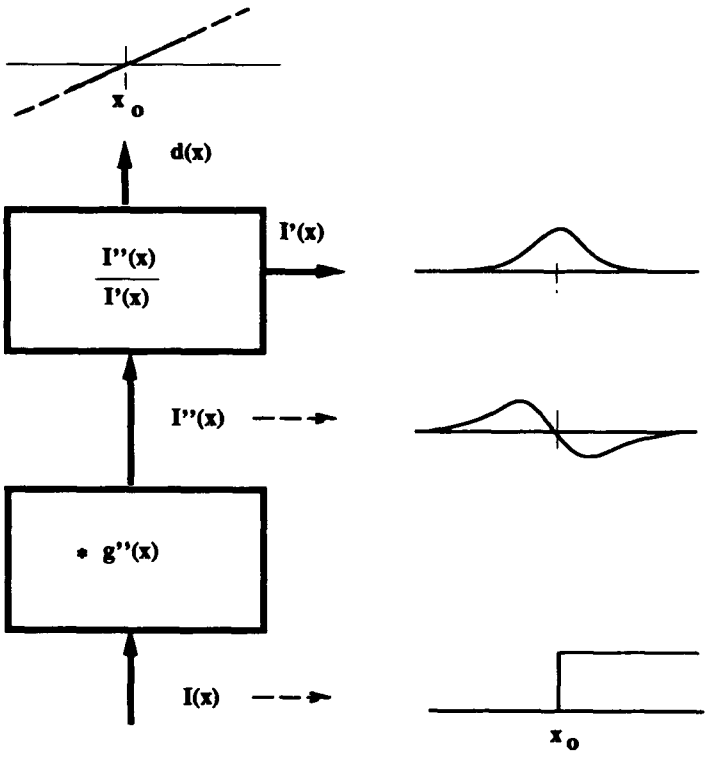


Figure 1: Basic 1D Displacement Function

pixel into something like $O(\sigma_b)$, it renders the local weighted estimate near the feature position x_o optimal. Since convolution is a linear operation, as are the operations which derive Disparity from Displacement representations, this convolution step can be deferred until after the Disparity computation in the present model to improve efficiency.

Thus the two functions $d(x)$ and $I'(x)$ encode the distance and contrast of proximate features in the 1D image. Blur is encoded in the displacement representation by its derivative — a sharp edge will result in a slope of σ_b^2, blurring will produce a lesser slope. Orientation will require the 2D model discussed later.

2.3 1D Stereo Disparity Model

As mentioned earlier, linear combinations of the Displacement function yield representations which should prove useful for such vision tasks as stereo disparity, optical flow, and object recognition. This paper focuses on stereo as an application domain.

In stereo, two viewpoints of an object can provide unambiguous depth information only if the correspondences between image points can be determined. This correspondence is a partial function mapping some points in the right image to points in the left. The difference between these positions is the disparity, and uniquely determines the distance of the feature in front or behind the fixation point.

In the case of the Displacement model, stereo disparity is determined by subtracting the left and right Displacement functions:

$$\begin{aligned} D(x) &= d_r(x) - d_l(x) \\ &= (x - x_r) + (x - x_l) \\ &= x_l - x_r. \end{aligned} \tag{4}$$

This approach imposes the constraint that the Displacement functions for both eyes at all x points in the domain represent the distance to corresponding features. This can only be completely true when the disparity is zero. On the other hand, when disparity is nonzero, at some scale, the features are proximate, and the disparity measure D will correspond to like features in the image and thereby give the correct result. This is the concept behind the scale space approach used in the stereo model. Also, unlike features will usually display differences in contrast, contrast sign, focus, and orientation (in 2D). All of these can be tested to reject false correspondences.

When matching feature Displacement functions are subtracted, the resulting Disparity is constant in the neighborhood of the estimate (see Figure 2). If the sharpness of the edges are mismatched, the slope of the Disparity will be nonzero. The absolute magnitude of slope of the Disparity also is constrained to be less than unity between any two features. Disparity slope, therefore is a useful constraint on the matching process [7].

In regions of the image where little or no feature information exists, such as areas of near constant intensity, little depth information can be inferred without imposing (top down) constraints on the solution. Marr and

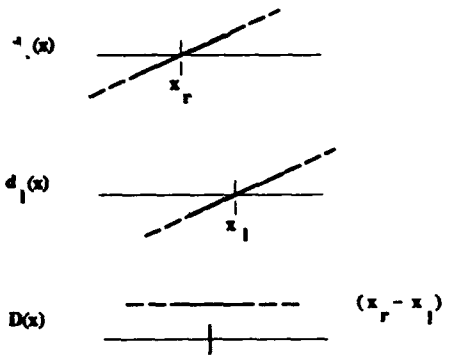


Figure 2: Basic 1D Stereo Disparity Function

Poggio [6] argued that the image should exhibit "smoothness" which has been exploited to handle such pathological cases. Any measure of stereo disparity must account for these pathological situations. In this model, $I'(x)$ when convolved with a Gaussian is taken as the inverse variance of the Displacement Function. Since Disparity is a linear combination, the variance of the estimate will be the sum of the individual input variances. This variance estimate is useful in avoiding featureless regions in the domain.

The model, as described so far, can account for feature position, contrast, and focus. It also has a means for rejecting spurious estimates. The full 2D model adds the feature orientation measure.

2.4 The General 2D Displacement Model

Happily, the 2D edge can be represented in such a way that the analysis is still essentially 1D. First we redefine a primitive edge function as:

$$\begin{aligned} d_o(x,y) &= x\sin\theta + y\cos\theta - d_o \\ I(x,y) &= \alpha u(d_o(x,y)) \end{aligned}$$

Note that we have merely added the orientation of the edge θ to the distance function $d_o(x,y)$.

As with the 1D model, the image is convolved with the Laplacian of a Gaussian (or a Gaussian convolved image is processed with a Laplacian operator):

$$\begin{aligned} I''(x,y) &= \nabla^2 g(x,y) * I(x,y) \\ &= \alpha g'(d_o(x,y)) \\ &= -\alpha \frac{d_o(x,y)}{\sigma_b^2} g(d_o(x,y)) \end{aligned}$$

Note we are still able to express this as a 1D Gaussian function of the distance to the edge $d_o(x,y)$.

Finally, instead of integrating this representation, as we did in the 1D model, we will take the gradient of the Gaussian convolved image. In cortex, it is probable that the Gradient representation (or something very much like it) is derived directly from the symmetric Laplacian-like center surround representation found in the retina and the Lateral Geniculate Nucleus. From a computational standpoint, however, this issue is not central to this discourse. Calculation of the gradient directly from the smoothed image is not a major computational challenge.

$$\begin{aligned} \mathbf{I}'(x,y) &= \nabla(g(x,y) * I(x,y)) \\ &= \alpha g(d_o(x,y))(\sin\theta\hat{\mathbf{i}} + \cos\theta\hat{\mathbf{j}}) \\ &= \alpha g(d_o(x,y))\hat{\Theta} \end{aligned}$$

From these two representations, we can arrive at a Displacement estimate of $d_o(x,y)$ just as we did in the 1D case, except we can choose between a vector and a scalar format:

$$\begin{aligned} d(x,y) &= -\sigma_b^2 \frac{I''(x,y)}{\|\mathbf{I}'(x,y)\|} \\ &= d_o(x,y) \\ \mathbf{d}(x,y) &= -\sigma_b^2 \frac{I''(x,y)\mathbf{I}'(x,y)}{\|\mathbf{I}'(x,y)\|^2} \\ &= d_o(x,y)\hat{\Theta} \end{aligned}$$

The above vector and scalar representations of Displacement functions can yield Disparity representations through subtraction or differentiation, just as was done in the 1D case. With the vector form $\mathbf{d}(x,y)$, the subtraction of two representations results in a Disparity vector field where the magnitude is proportional to the shortest distance between the two features, and the direction is normal to the features (assuming aligned features).

This form of measuring Displacement is ideal for matching and motion in the 2D scene. For stereo, however, we are interested in a more restricted measure of feature positions — epipolar Displacements.

2.5 The 2D Stereo Displacement Model

A stereo image pair has the constraint that for any image point in one scene p_l, there exists a one dimensional slice on the other image e_r which must contain the corresponding point, assuming it exists. Conversely any point along that corresponding line in the second image p_r is constrained to match to a point along a line e_l in the first image through the point P_l. This is a constraint imposed by the geometry of the optical setup. These lines, called epipolars, also lie on a common plane which passes through the imaged point as well as the focal points of the cameras (See Figure 3).

Another aspect of the stereo configuration related to the alignment of edges to the epipolar lines. When features are normal to these basically horizontal lines, the variance of the displacement measure along the epipolar is as stated before; inversely proportional to the gradient denominator $I'(x)$. As features are rotated such that they are oblique to the epipolars, the variance on feature displacement estimation increases to the point where, when features are aligned with the epipolar, there is absolutely no way to measure the intersection point of the feature and the epipolar.

The correct weight, or inverse variance, for an epipolar Displacement can be calculated from the Displacement

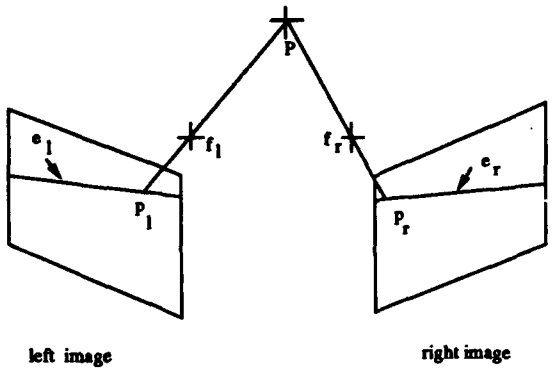


Figure 3: Stereo Epipolar Constraint

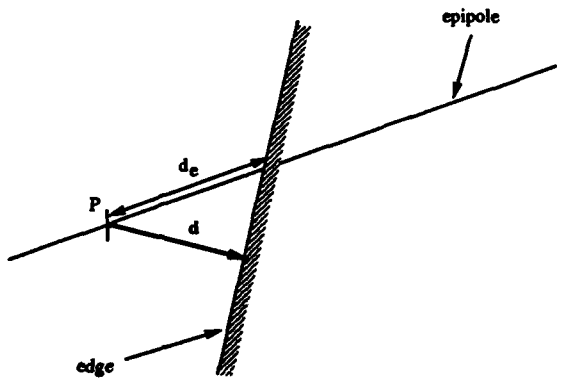


Figure 4: Epipolar Displacement d_e

weight, $\mathbf{I}'(x,y)$, and the epipolar direction vector at the image point $\hat{\mathbf{e}}$:

$$\begin{aligned} w_e(x,y) &= \mathbf{I}'(x,y)\cdot\hat{\mathbf{e}}(x,y) \\ &= \alpha g(d_o(x,y))(\hat{\Theta}\cdot\hat{\mathbf{e}}(x,y)). \end{aligned}$$

A similar modification must be made to the 2D Displacement representation, since a small normal displacement $\mathbf{d}(x,y)$ will result in large displacements along the epipolar when the feature is oblique to it (See Figure 4):

$$\begin{aligned} d_e(x,y) &= -\sigma_b^2\frac{I''(x,y)}{w_e(x,y)} \\ &= \frac{d_o(x,y)g(d_o(x,y))}{w_e(x,y)} \\ &= \frac{d_o(x,y)}{\hat{\Theta}(x,y)\cdot\hat{\mathbf{e}}(x,y)} \end{aligned}$$

The epipolar Displacement, therefore, is similar to the 1D Displacement in being a scalar function, but is based on the relationship between the 2D Displacement vector function $\mathbf{d}(x,y)$ and the epipolar field $\hat{\mathbf{e}}(x,y)$.

It is, perhaps, useful to note that unlike many stereo algorithms, this approach requires no restriction on the epipolar configuration or other imaging alignment assumptions.

3 The Scalespace Stereo Algorithm

The above sections outlined the derivation of the stereo displacement, i.e. the distance along any epipolar from any point on the epipole to the nearest feature. This is weighted by an inverse variance estimate w_e which accounts for the uncertainty associated with oblique feature alignments as well as low feature contrast. The blur and feature position are encoded in the displacement estimate d_e. All that remains is to combine them into a stereo Disparity estimator.

As before, the basic calculation involves simple subtraction of displacement estimates and summation of variance estimates:

$$D'(x,y) = (d_r(x_r,y_r) - d_l(x_l,y_l))/2 + D(x,y)$$
$$W(x,y) = \left(w_r^{-1}(x_r,y_r) + w_l^{-1}(x_l,y_l)\right)^{-1}$$

where (x_r,y_r) and (x_l,y_l) are positions along matched epipolars in the stereo image pair (a calibration issue) displaced by any prior Disparity estimate available — $D'(x,y)$. These choices for prior estimates of disparity along the epipoles is where the scale space approach enters in.

As mentioned earlier, there is a need to avoid false correspondences in any stereo algorithm. With this representation, however, we have substantial information on hand to help avoid such mistakes. Given an estimated Disparity, the algorithm looks up the positions along the appropriate epipoles (x_r, y_r, x_l, andy_l) and calculates the Disparity D and weight W above. Any or all of the following criteria can be used to detect mismatches and, if so, zero the weight function $W(x,y)$ over the offending intervals:

- If the contrast signs are mismatched (the signs of the w_e's don't match), or
- a real edge is being matched to an illusory edge [1], or
- the blurs are severely mismatched (resulting in a sloped D function near the feature), or
- the edges are misaligned, or
- the absolute Disparity or the Disparity slope is too large.

Once this filtering takes place, the functions W and D' are both convolved with a gaussian of width σ_b. This, in effect, generates the optimal variance estimate near the feature locations, as discussed earlier (see Equations 2 and 3). Also, this operation effectively interpolates between features and "fills in" where mismatches were detected and their erroneous disparities rejected:

$$D(x,y) = \frac{g * [D'(x,y)W(x,y)]}{g * W(x,y)}$$

The smoothed Disparity, then, is fed to the next smaller scale calculation. This determines the displacement along the epipolars for the sampling (x_r, y_r) and

[1] These illusion edges are referred to as *Chevruels* in the psycophysical literature and are characterized by inverse sloped disparity functions

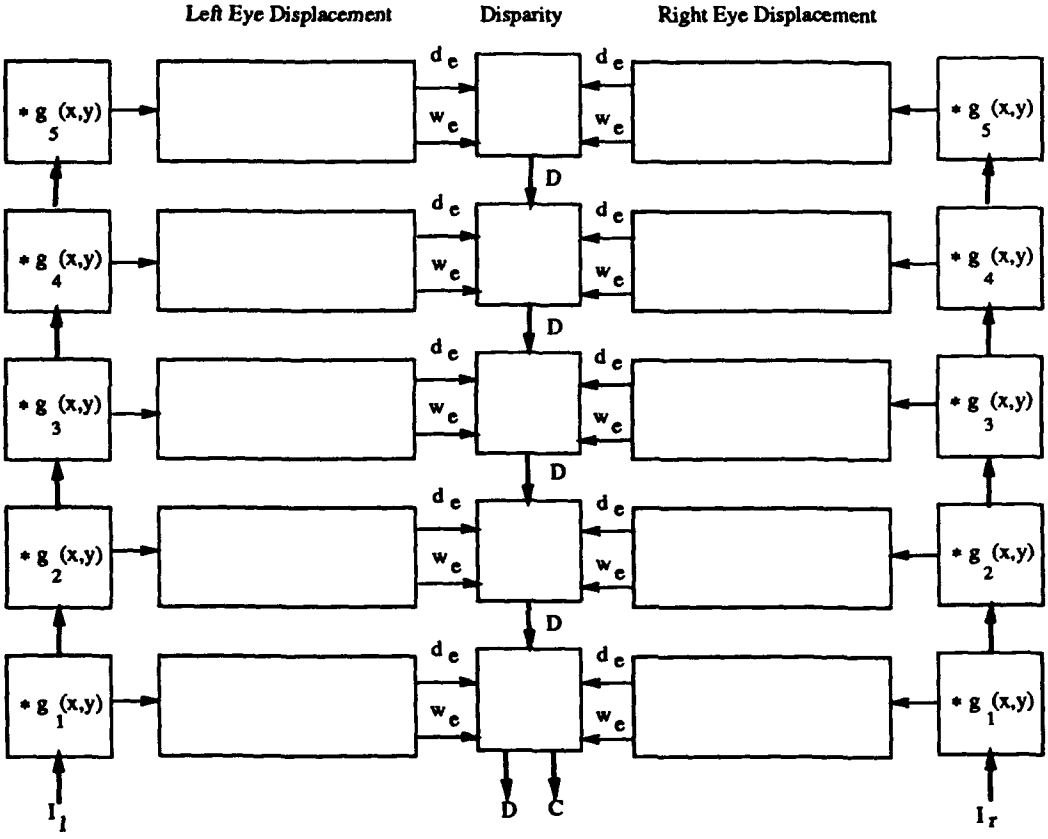


Figure 5: Stereo Scale Space Algorithm

(x_l, y_l). At the largest scale the initial Disparity estimate $D(x, y)$ is assumed to be zero. The weighting function $W(x, y)$, is not used further in this model, although it almost certainly could be incorporated in the scale space scheme.

At the smallest scale, however, the weight function is sampled where the "cyclopean" Displacement function has zero crossings. This function:

$$C(x, y) = (d_r(x_r, y_r) + d_l(x_l, y_l))/2$$

is, in effect, the fused binocular Displacement function. The Disparities $D'(x, y)$ associated with significant weight values $W(x, y)$ at zero crossings in the cyclopean Displacement $C(x, y)$ are precisely the sparse stereo Disparity representation desired.

4 Three Experiments

The algorithm described above, was run on many real stereo pairs. Three are presented here as representative of the algorithm's performance. The first, the "Jet and Decoy" demonstrates some of the capabilities of the approach in terms of acuity resolution. The second, the "Hallway" contains, in contrast, extremely large absolute disparities and disparity gradients. The last "UBC", has a large amount of detail and, by it's poor calibration, noise. It is a good test of the robustness of the approach in such situations.

It is important to note that many possible filters mentioned earlier, such as blur matching, edge misalignment, and disparity slope have *not* been incorporated into these experiments, since I felt it would be more interesting to see when such filters might be useful by allowing their absence to result in pathological correspondences. I shall show one such example below.

Also, it should be noted that, because of this hands-off experimental design, there were very few "knobs" in the algorithm. In all instances, no parameter was found to be either critical in it's selection or vary observably in its optimal setting between experimental runs. As such, basically arbitrary choices were made as to the number of scales and the scale width spacing (σ_b steps of 2.0 were used), absolute disparity measure at any scale ($\pm 3.0\sigma_b$), iterations at any scale (2). The latter was useful at the largest scales where no prior on $D(x, y)$ is available.

4.1 The Jet and Decoy

This image pair was taken of a model of a jet and a paper cutout using the MIT head/eye stereo camera system. The two images shown in Figure 6 are the left grey-scale image and an image with only the sparse depth estimates highlighted. Only the highest contrast features are shown. The toy jet, in the lower left corner, has a disparity of only 1-2 pixels larger than the paper decoy.

In the depth map, the Disparity readings are encoded as white being a Disparity of three pixels or more and

black being two pixels or less. In other words, the total dynamic range indicated for this picture is *one* pixel. Note that the model jet has all of its Disparity measures as three or more pixels with the exceptions of the shadows it casts on the ground and on it's own tail, whereas the decoy is clearly darker. It is even possible to discern the relative Disparity within the decoy, in spite of this one pixel total dynamic range.

4.2 The Hallway

In the second example (Figure 7), a corridor provides an environment which has very large absolute disparity as well as disparity gradients approaching the limit of 1.0. The disparity range of -20 pixels (black) to 10 pixels (White) is substantially larger than the Jet and Decoy example. In fact, an example of the disparity gradient limit being exceeded can be seen in the middle of the picture, where the only error correspondence takes place. In one image, the reflection of the overhead lamps is in the middle of the window and in the other (shown) they abut the central door edge. In this instance, along with the "bloom" associated with the reflected image, there is no like contrast edge to match at the window edge, and the matcher (incorrectly) matched it to the edge between the doors.

A simple hack could rule this out (the gradient exceeds 1.0 between neighboring features) but I thought it more interesting to leave out any gradient filters to test precisely when such a filter might be useful. It is important to note that, despite the ample opportunity for false correspondences of this sort in any of these images, it is only when the geometry and optics of the setup created such an extreme violation of the gradient limit that such a failure is found.

4.3 The Campus

Figure 8 is the depth map made from a stereo pair taken from an aerial shot of the University of British Columbia campus. It is a substantially denser depth map than the previous examples. The minimum weight $W(x,y)$ was chosen such that about 10,000 estimates are plotted.

This example is illustrative of the substantial detail achievable with the algorithm without significant spurious feature content. The upper left region is all labeled with lighter depth markings due to the depressed elevation. The buildings in the central region are distinguished by their high (dark) elevation markings. Note the small rectangular place in the middle of the central building. The depression in the disparity map approaches the theoretical disparity gradient limit of 1.0.

An artificial elevation slope in the image is caused by a slight rotation between the two images. This could be calibrated out using the epipole tables, if desired.

5 Conclusion

A very simple Displacement representation was derived from the Laplacian and gradient of a Gaussian convolved image which provides, at any image point, a measure of feature proximity, contrast, orientation, and blur. Simple subtraction of the Displacement function renders a Disparity measure which is useful for higher level processing.

Stereo differs from other vision modalities in that a constraint can be imposed both on the matching space (the epipolar constraint) and on the maximum gradient in the disparity map (1.0). The epipolar constraint can be easily integrated into the the Displacement representation. Other constraints can be imposed to reduce false matches.

In experiments, however, the need for further constraints arises infrequenty. With the exception one uncorrected violation of the disparity gradient created by a conspiracy of the physical setup and the optical behavior of the imager, as well as a few miscorrespondences due to incorrect assumptions about the alignment (calibration) of the images, no other errors were found in these or other experimental runs. Including all sources, the overall error rate never exceeded 0.1% of the total feature count.

In addition to the foregoing work in stereo, continuing research indicates that the Displacement model of early vision processing may hold considerable promise in many other areas such as motion, recognition, and even in the modeling of early cortical processing in primates.

References

[1] A. Barr and E. Feigenbaum, *The Handbook of Artificial Intelligence*, Vol. 3, 216-223, Houristech Press, Los Altos, Ca. (1981)

[2] J.F. Canny, "Finding edges and lines in images," MIT A.I. Tech Report TR-720, (1983)

[3] W.E.L. Grimson, *From Images to Surfaces: A Computational Study of the Human Early Visual System*, The MIT Press, Cambridge, Ma. (1981)

[4] E. Hildreth, "Edge detection," MIT A.I. Memo No. 858 (1985)

[5] B.K.P. Horn, *Robot vision*, Chapter 8, The MIT Press, Cambridge, Ma. (1986)

[6] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," *Science*, **194**: 283-287 (1976)

[7] S.B. Pollard, J.E.Mayhew and J.P. Frisby, "PMF: A stereo correspondence algorithm using a disparity gradient limit" *Perception*, **14**: 449-470 (1985)

[8] T. Sanger, "Stereo disparity computation using Gabor filters," *Biological Cybernetics, 59*: 405-418 (1988)

[9] V. Torre and T. Poggio, "On edge detection," IEEE Trans. PAMI-8, No. 2: 147-163 (1986)

[10] M. Turk and A. Pentland, "Eigenfaces for recognition," *MIT Media Lab Vision and Modeling Group, Technical Report*, No. 154, (1990)

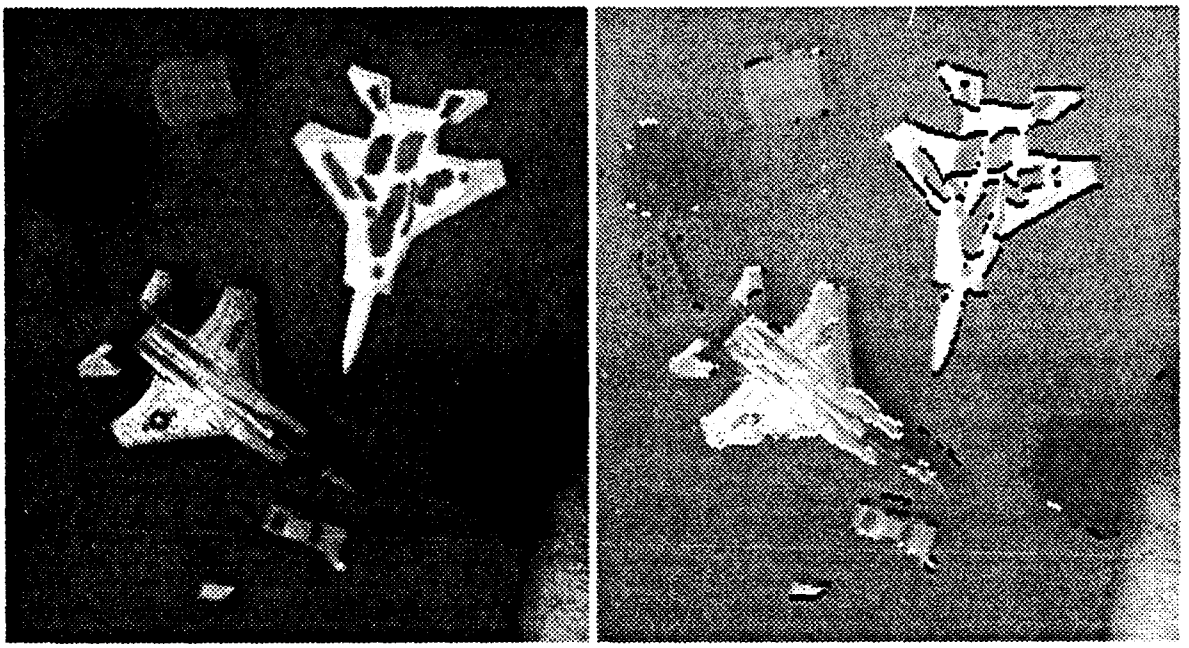


Figure 6: Jet and Decoy — Image and Depth Map

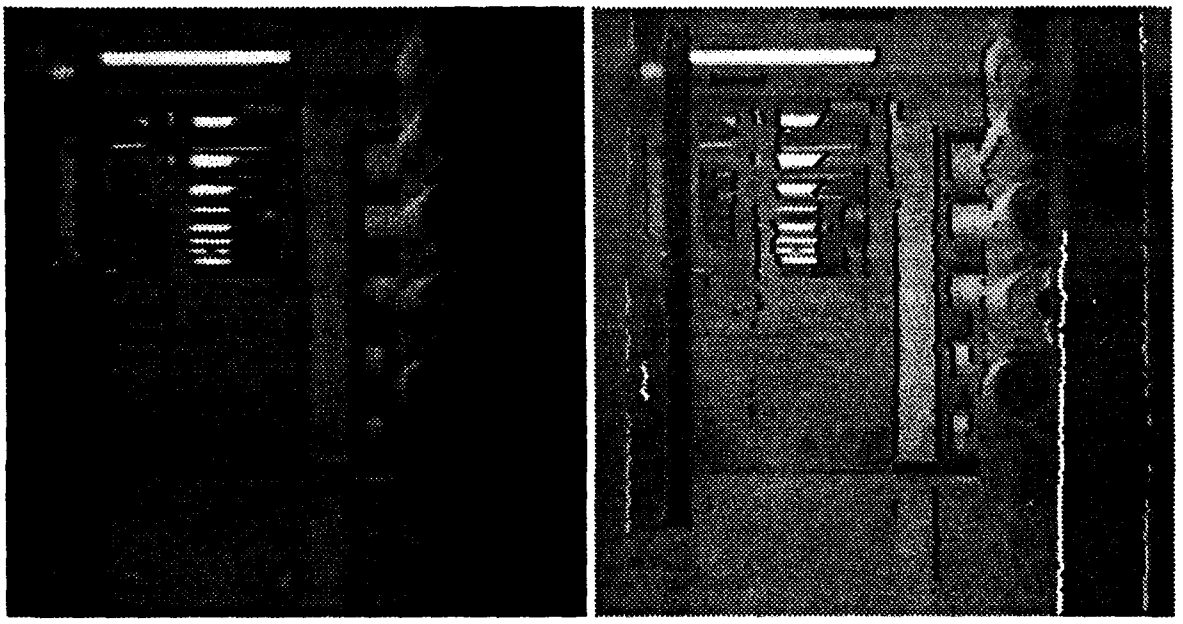


Figure 7: Corridor — Depth

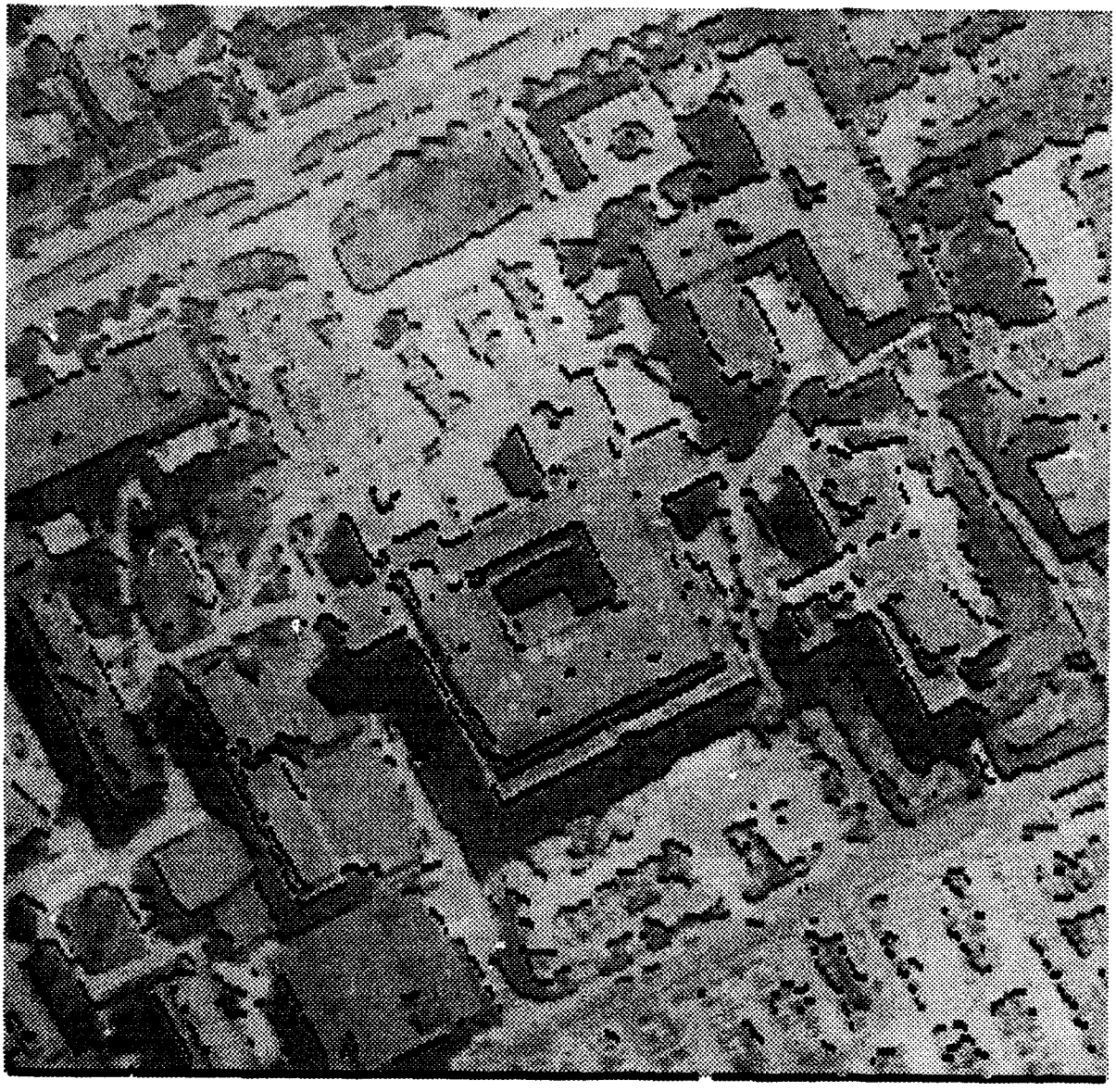


Figure 8: UBC — Depth

RECOVERING LSHGCs AND SHGCs FROM STEREO *

Ronald C.-K. Chung and Ramakant Nevatia

Institute for Robotics and Intelligent Systems
Departments of Computer Science and Electrical Engineering
University of Southern California
Los Angeles, CA 90089-0273

email: rchung@iris.usc.edu, nevatia@iris.usc.edu

Abstract

We examine the problem of computing volumetric shape from stereo. We argue that intermediate $2\frac{1}{2}$-D dense or wire-frame descriptions may not be always possible from stereo, especially when there are curved surfaces in the scene, and that 3-D volumetric descriptions of objects may have to be derived directly from stereo correspondences. We then present methods to recover volumetric shape using LSHGCs and SHGCs as the shape models, based on some invariant properties in their monocular and stereo projections. Experimental results on both synthetic and real images of objects with curved surfaces are given.

1 Introduction

One basic goal of vision is the capability to extract descriptions about shapes of objects in a given scene, either for storing into the memory if the objects are new, or for comparing with descriptions of known objects to achieve recognition. Use of stereo is common to recover 3-D descriptions of a scene by using multiple images. Yet features such as corners and line segments extracted from images and possibly matched in stereo are generally too local for indexing purposes during recognition. The alternative is to use more global descriptions such as surface and volumetric descriptions which have more discriminative power for indexing. The question is, how can we recover such shape descriptions from stereo images without referencing to specific object models?

Traditionally, stereo has been thought of as a process merely to build a $2\frac{1}{2}$-D dense depth map of a scene, and separate from the process of global shape description which comes later. Yet in cases where the objects in the scene are not densely textured, most of the features that can be extracted and matched in stereo are merely edges along surface boundaries. Surface interpolation from such boundary conditions alone is generally difficult. Some might argue that dense surface descriptions may not be necessary; even wire-frame descriptions of surfaces are very useful in many applications, and we can infer volumetric descriptions from them if necessary. However, as we shall see in the following, even depth measurements along surface boundaries may not be always directly available from stereo.

It has long been known that the apparent boundaries of a curved surface at its limbs in an image are viewpoint-dependent; the boundary edges in the image are projected from different contours on the surface with different angles of view. As a result, the apparent boundaries of a curved surface in its stereo images may not actually correspond. This is illustrated in figure 1. In the following we call those contours on the surface which project to image contours *contour generators*, and the image contours projected from the limbs of curved surfaces *limb edges*.

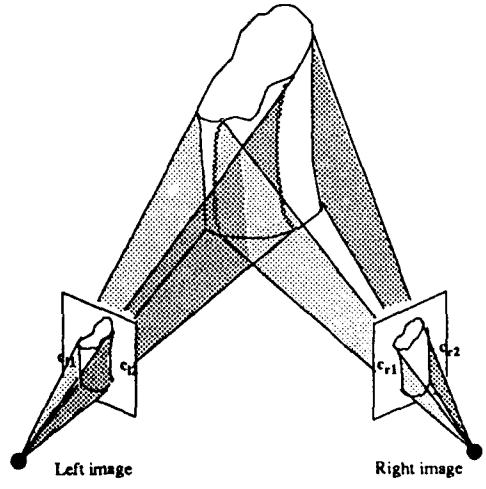


Figure 1: Stereo projections of a curved surface.

This is a unique problem of using multiple intensity images for measuring depth, and is basically what makes deriving shape from stereo different from, and more difficult than, that from range images. This limb problem only gets worse as the image resolution or the stereo angle is increased in an attempt to get higher accuracy in the depth estimates. Worse yet, if this issue about

*This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-90-C-0078. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

curved surfaces is not addressed, the surface descriptions of the scene may be mistaken in taking limb edges as real edges.

In an earlier work [3], we have proposed a stereo system which uses high level structural descriptions for stereo correspondence. Hierarchical descriptions up to surface level are computed from each image using a perceptual grouping technqiue based on symmetries [6; 5], and such descriptions in the two images are used for stereo correspondence. The output of such a system is therefore not merely depth estimates along edges in the scene, but also segmented surfaces. The system is also able to distinguish limb edges from real edges by observing the behavior of the junctions at the ends of the edges. However, it did not recover 3-D information along those limb edges. This paper describes how depth information along those limb edges can be estimated from stereo as a by-product of the shape description process.

Since dense $2\frac{1}{2}$-D depth measurements are not always available from stereo correspondences, we propose to infer volumetric descriptions not via intermediate dense depth measurements, but rather *directly from stereo correspondences*. Moreover, as there are generally infinite number of volumetric shape that can project to the same sparse scene data, we need to make use of some shape models that are common and give results that are consistent with human perception. We propose to use Generalized Cones (GCs), first introduced by Binford [1], as the shape primitives for the reconstruction. This paper concentrates on how Linear Straight Homogeneous Generalized Cones (LSHGCs) and Straight Homogeneous Generalized Cones (SHGCs) can be reconstructed from stereo images. The key to the solution is the use of some invariant properties of LSHGCs and SHGCs in both their monocular and stereo projections. Our technique can be summarized as follows: first hypothesize LSHGCs and SHGCs from the images using some invariant properties of their 2-D projections; then establish correspondences among the image contour points across the stereo images using the known epipolar geometry; finally reconstruct volumetric descriptions from the correspondences.

We first outline in section 2 some previous work on reconstructing curved surfaces and inferring volumetric descriptions from stereo. We describe how LSHGCs and SHGCs can be reconstructed from stereo and present some experimental results in sections 3 and 4 respectively. Finally we give the conclusion in section 5.

2 Previous Work

Rao and Nevatia [9; 8] described a technique of deriving volumetric descriptions of LSHGCs from stereo. Their approach employs a hypothesis-and-verification strategy, using the following two properties of LSHGCs originally given in [10]: (1) The contour generators of a LSHGC are two straight line segments; (2) The contour generators of a LSHGC are coplanar. The system takes an input of possibly sparse and imperfect $2\frac{1}{2}$-D depth measurements from stereo. An LSHGC is either hypothesized from its contour generators exhibiting the above properties and verifed by the cuts which they call *terminators* at the two ends of the LSHGC, or the reverse.

The proposed methods, however, make the assumption that $2\frac{1}{2}$-D depth measurements along the contour generators are available, which is clearly not the case along the limbs of curved surfaces as described earlier.

Lim and Binford [4] were the first to explicitly address the problem caused by limb edges in the process of reconstructing curved surfaces from stereo. They did not indicate how limb edges can be identified, but they proposed a method to reconstruct the curved surfaces from stereo. The object in the scene is assumed to be composed of a number of parallel cross-sections, each of them can be described by a conic function which involves five parameters. The curved solid is first cut into a number of slices such that each slice is in an epipolar plane. Within each epipolar plane, the four lines of sight can be recovered from the stereo images, and the cross-section is a conic which is tangential to the four lines of sight. This leaves only one free parameter for the conic in each epipolar plane. Two constraints, the extremum constraint and the terminator constraint, have been proposed and either of them can be used to determine the free parameter. The extremum constraint chooses the most compact shape among all possible conics in each epipolar plane. The terminator constraint chooses the conic which has the same eccentricity as that of the boundary of the terminator surface.

There are still problems with Lim and Binford's method: (1) Epipolar planes are parallel to each other only when the projection geometry is orthographic. However, the limb problem is significant only when the curved surface is at close range where orthographic approximation is not a good one. This means epipolar slices, being non-parallel to one another, generally do not possess similar characteristics and neither terminator constraint nor extremum constraint can be applied to them as a whole. (2) Epipolar planes are in general not parallel to the terminator surface as shown in figure 2. The difference in the orientations strictly depends upon the orientation of the solid with respect to the cameras. This implies the epipolar slices have no direct relationship with the terminator surface and thus terminator constraint is generally not applicable. On the other hand, applying extremum constraint will give a stack of conics being most compact only in the orientations of the epipolar slices. The shapes recovered therefore will be different with different angles of view of the solid.

3 Shape Recovery of LSHGCs

To reconstruct LSHGCs from stereo, we first need to look at a number of properties of LSHGCs in the images. We present the properties and propose a method for the reconstruction in section 3.1. Experimental results then follow in section 3.2.

3.1 The Method

An LSHGC is a volume defined by sweeping a given cross-section function along a straight line called axis, such that the cross-section is scaled linearly along the

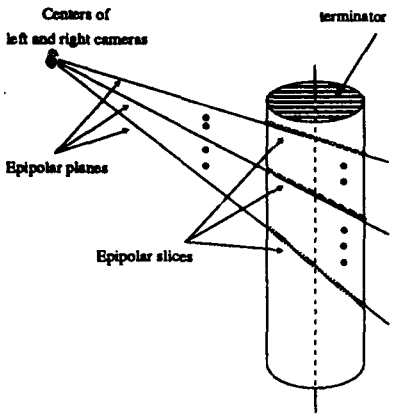


Figure 2: Lateral view of the stereo camera geometry. Epipolar planes in general are neither parallel to one another nor parallel to the terminator surface of a solid.

axis. Linking the points on the surface which correspond to the same unscaled arc length s along boundaries of the cross-sections, we have *meridians*. Because of the linearity in the sweeping function, all meridians are straight and intersect at a point which we call *apex point* on the axis, at which the scaling is zero.

Lemma 1 *The image contour of an LSHGC is the projection of one of its meridians under orthographic or perspective projection.*

Proof Shafer and Kanade [10] have algebraically shown that the contour generator of an LSHGC is a straight line under orthographic projection. Here we give a more intuitive proof that the contour generator is in fact one of the meridians *regardless* of the projection geometry. Because of the linearity in the sweeping function, the surface normals to the LSHGC surface at points along a meridian are all parallel and perpendicular to the meridian. Say a line of sight from the optical center C of a camera touches the LSHGC surface at point P as shown in figure 3. Let m_P be the meridian passing through P. Then both CP and m_P are perpendicular to the surface normal at point P and they define the tangent plane π to the surface at point P. Since all surface normals along a meridian are parallel, the tangent plane π is orthogonal to all the surface normals along the meridian m_P, and so is any line on the plane π. As a result any point Q on the meridian is a point on the contour generator since the line of projection CQ lies on the plane π. □

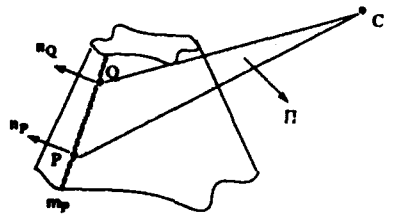


Figure 3: Contour generator of an LSHCC is a meridian.

Theorem 1 ***Given four image contours of an LSHGC in a stereo pair of images, the points of intersection among their extensions in the two images are projections of the apex point of the LSHGC and fall on corresponding epipolar lines under orthographic or perspective projection (An example illustrating the theorem is shown in figure 4).***

Proof Since all meridians intersect at the apex point in 3-D, the projection of meridians to any image will also intersect at the projection of the apex point to that image. By lemma 1 the image contours in the stereo images are projections of meridians; their extensions therefore intersect at the images of the same point in 3-D, the apex point. As a result the points of intersection fall on corresponding epipolar lines. □

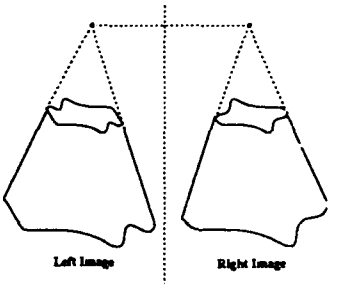


Figure 4: Stereo correspondence of LSHGC contours.

Theorem 1 not only allows us to hypothesize an LSHGC from stereo images, but also to recover the apex point in 3-D simply by matching the image apex points projected by the image contours. Notice that a contour in 3-D on the surface of the LSHGC can be recovered by matching the terminator contours in stereo. The apex and the 3-D contour therefore uniquely define an LSHGC which projects to the four image contours, regardless of how the cone is cut at the two ends. If the cuts are important, we can first recover their partial descriptions in 3-D by matching their images, and the cuts are where the partial descriptions intersect with the cone in 3-D.

Since we have not specified any particular cut on the LSHGC, the method works even for cuts being non-planar or non-orthogonal to the axis.

3.2 Experimental Results

Results on synthetic images of an LSHGC are shown in figure 5. We begin with a stereo pair of intensity images and compute the edges using Canny's edge detector [2]. We then compute the hierarchical descriptions from each image using a perceptual grouping technique [6; 5], and match those descriptions in stereo [3]. The output of such processes is segmented and matched surfaces in stereo. Junctions are also labelled as either limb-junctions or real junctions, and from them limb edges are also identified. Finally we determine the image apex points of the LSHGC by extending the limb edges, and recover the volumetric descriptions of the object in the scene.

4 Shape Recovery of SHGCs

Ulupinar and Nevatia [11] have recently demonstrated that the shape of an SHGC can be recovered merely from a single line drawing. As the problem is highly underconstrained with one view, the method has to make

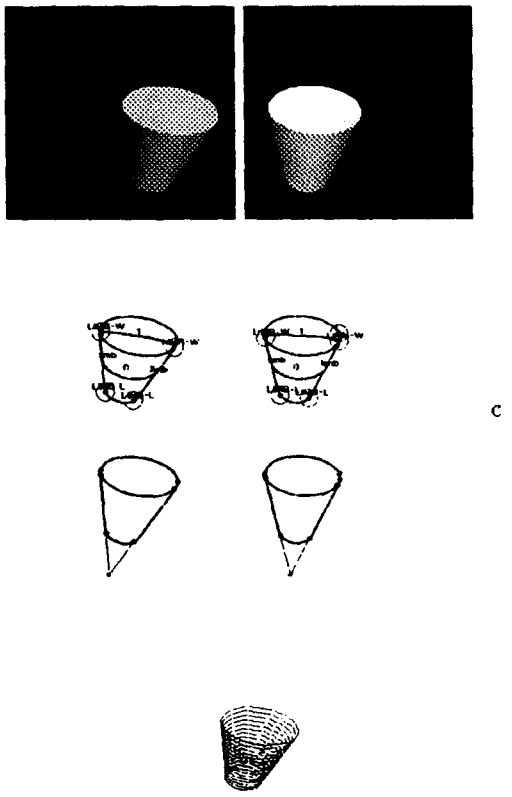


Figure 5: Results for a synthetic scene of an LSHGC (from top to bottom): the left and right images; the left and right matched ribbons and identified junctions; the left and right extracted image apex points; the recovered volumetric descriptions projected to the left view.

use of some perceptual assumptions which are believed to give results consistent with human perception. It is also restricted to using orthographic projection as its projection model.

In the following we show that if we have a second "eye", there is a more direct method to compute the shape as well as the pose of an SHGC without making those perceptual assumptions. In addition, our method can also handle images projected under perspective projection. We do assume that the cross-section function is visible from at least one of the cuts of the SHGC. However, we do not require the cross-sections to be orthogonal to the axis, *i.e.*, oblique SHGCs are allowed.

4.1 Hypothesizing SHGCs

To reconstruct SHGCs in a scene, we first have to find evidence for their existence in the images. Ponce *et al.* [7] have shown an important theorem concerning the tangents to the image contours of an SHGC under orthographic projection. Ulupinar and Nevatia [11] extended it to perspective projection and give a more intuitive proof for the theorem. The theorem can be stated as this: *If two image contour points of an SHGC are from the same cross-section, the tangents to the contours at these points when extended intersect on the projection of the axis.* Using this property of SHGCs, we can hypothesize the existence of an SHGC by establishing pairwise correspondences between points on the image contours in each image such that their tangents intersect on the same straight line, with the straight line so derived being the projection of its axis.

The pairwise correspondences and the projected axis in each image can be estimated using Hough Transform as in [7], and confirmed by checking whether all corresponding pairs of points follow the same order and are continuous along the contours.

After we collect enough strong evidence of the existence of an SHGC, the next step is to reconstruct it. The idea is simple. Matching the projections of the axis in the stereo images will automatically recover the 3-D position of the axis. Similarly, matching the projections of the terminator boundaries will recover the 3-D shape of the cross-section function. What is left is to scale the cross-section function in 3-D so that it touches the lines of projections from the same cross-section of the solid.

The problem is that, we have to first set up correspondences across the stereo images so that we know which pair of points in the left image and which pair of points in the right image are projected from the same cross-section. This is nontrivial as in case if the contour generators are limbs, the corresponding points in the images are indeed projections from four different points on the surface of the SHGC. Here let us call the corresponding points in the stereo images projected from any given cross-section p_{l1}, p_{l2}, and p_{r1}, p_{r2} respectively.

4.2 Setting Up Correspondences across Stereo Images

The correspondence problem is simpler when the contour generators are creases instead of limbs, in which case p_{l1} and p_{r1} are projections of the same point in 3-D and will fall on corresponding epipolar lines, and so will p_{l2} and p_{r2} (see figure 6). This shows the importance of identifying the nature of the visible surface boundaries as being creases or limbs.

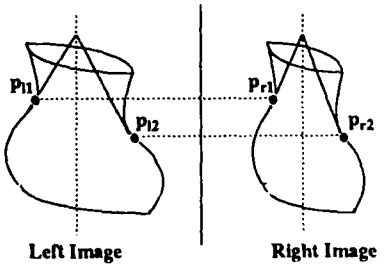


Figure 6: Corresponding points on the image contours of an SHGC which belong to the same cross-section, when the contour generators are creases.

To solve the correspondence problem when the edges are limbs edges, we first examine a number of properties of SHGCs in their projections. Let us first define the tangent to a surface at a point P *in the direction of* a line L in 3-D be the tangent which lies on the plane containing the point P and the line L.

Lemma 2 (Shafer and Kanade [10]) *Given points on the surface of an SHGC that belong to the same cross-section, the tangents to the surface at these points in the direction of the axis when extended intersect at a*

common point on the axis (see figure 7).

Following Shafer and Kanade, we call the common point of intersection on the axis the *apex point*, and the tangents in the direction of the axis the *apex tangents* of the given cross-section. We also call the 2-D projections of the apex point and the apex tangents on any image plane the *image apex point* and *image apex tangents*. Notice that different cross-sections of an SHGC generally have different sets of apex tangents and different apex points on the axis.

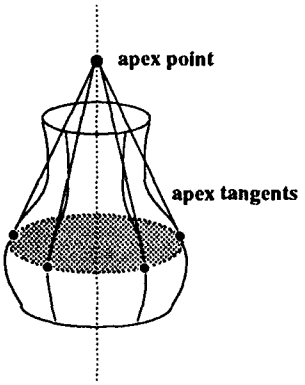


Figure 7: The apex point of a given cross-section of an SHGC.

Ulupinar and Nevatia [11] have also discovered a property of the limb edge of any curved surface:

Lemma 3 (Ulupinar and Nevatia [11]) *All the tangent lines to a surface at a point, say P, which is on a limb edge of the surface under any given projection geometry, project as the same line on the image plane.*

This property, in combination with lemma 2, imply that tangents to the limb edges at points which belong to the same cross-section, say p_{l1} and p_{l2}, are in fact equivalent to the 2-D projections of the apex tangents at those points. As a result their point of intersection in the image will also be the 2-D projection of the point of intersection of the apex tangents. This has been included in the work of [7] and [11] when they prove that tangents to the limb edges intersect on the projection of the axis. We rephrase it as in below:

Lemma 4 (Ponce *et al* [7], Ulupinar and Nevatia [11]) *Given two points on the limb edges of an SHGC that belong to the same cross-section, the point of intersection between the tangents at those points is the image apex point of that particular cross-section.*

Combining lemmae 2 and 4, we get the following theorem:

Theorem 2 *If four points on the limb edges of an SHGC in a stereo pair of images belong to the same cross-section, the points of intersections among the tangents at those points in the two images fall on corresponding epipolar lines (An example illustrating the theorem is given in figure 8).*

Proof By lemma 4 tangents at the image points intersect at the image apex point of that cross-section in each image. Since the four points are from the same cross-section, and by lemma 2 apex point is uniquely defined for each cross-section, the two image apex points in the stereo images are in fact projections of the same point. As a result the two image apex points fall on corresponding epipolar lines. □

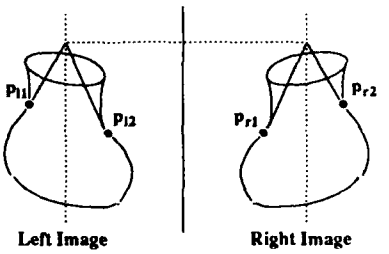


Figure 8: Corresponding points on the image contours of an SHGC which belong to the same cross-section, when the contour generators are limbs.

Theorem 2 gives one way to establish correspondences among points on the limb edges which belong to the same cross-section. What is remaining is to scale the cross-section function in 3-D to project to the four corresponding points. This is still nontrivial as the four points correspond to four lines of projection which in general are not coplanar. The following describes a non-iterative method of how we can compute the cross-section in 3-D for each set of correspondences. Remember that the cross-section function in 3-D can be computed by matching the terminator contours in stereo. Similarly, the axis of the SHGC in 3-D can be recovered by matching the image axes.

4.3 Fitting the Cross-Sections

We have obtained for every cross-section slice of the SHGC two corresponding image apex points and four image apex tangents in the stereo images. Here we treat the recovery problem as one to recover a virtual LSHGC whose apex is the apex point of the cross-section, whose meridians are the apex tangents of that cross-section, and whose cut is the cross-section itself. Following the scheme in section 4, we first derive the LSHGC and then determines the proper cut that coincides with the scene data.

By matching the image apex points we can recover the apex A of the cone (see figure 9). We then move the cross-section function down the axis in 3-D to some distance, say t from the apex. We call the new axis point $C(t)$. Pick one of the four image apex tangents, say the one at point p_{l1}. The image apex tangent and the optical center of the corresponding camera form a plane of projection which we call π, to which the virtual cone should touch. For each point s along the boundary of the cross-section function, we compute two measures: the distance $r(s)$ from $C(t)$ to the boundary point s, and the distance $R(s)$ from $C(t)$ through the point s to the plane π. The fraction $R(s)/r(s)$ is the scaling of the cross-section such that the point s touches the tangent plane π. The proper scaling of the cross-section function to touch with the plane π, regardless of whether the edges are limb edges or real edges, can therefore be

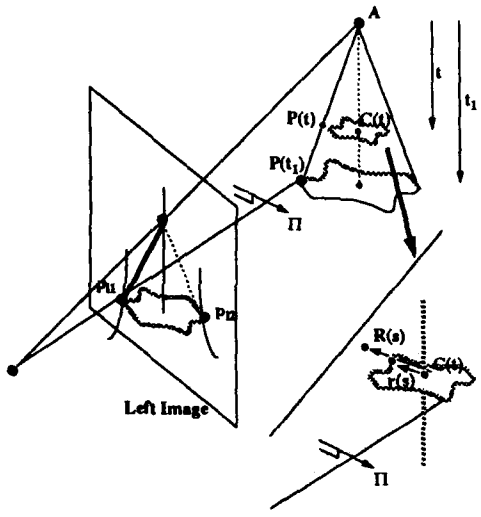


Figure 9: Recovering a cross-section.

computed as:

$$scale(t) = \min_s R(s)/r(s)$$

The apex point A, the axis, and the scale function $scale(t)$ at distance t along the axis uniquely define a virtual LSHGC which gives rise to the image apex tangents.

The next step is to recover the proper cut of the cone to project to the four corresponding points. From the above process we can easily recover the point of contact $P(t)$ between the projection plane π and the scaled cross-section of the cone at distance t from the apex. The line $AP(t)$ then defines the contour generator on the cone which projects to the given image apex tangent. Notice that the contour generator of an LSHGC has to be a straight line. Let us say that the proper cut is at a distance t_1 from the apex along the axis, and the proper scaling of the cross-section function on that plane is $scale(t_1)$. The point $P(t_1)$ on the surface of the cone which projects to point p_{l1} is then given by the intersection of two lines: the line of projection through point p_{l1}, and the contour generator $AP(t)$. Finally, using property of similar triangles we can get:

$$t_1 = (AP(t_1)/AP(t)).t$$
$$scale(t_1) = (AP(t_1)/AP(t)).scale(t)$$

The above cone and cut recovery processes can be applied to any one of the four image apex tangents. In principle they should all return the same cross-section, unless the object is in fact not an SHGC but merely looks like an SHGC, or the cut being used as the cross-section function is actually not along one of the cross-sections. This serves as an additional verification from the reconstruction process. In case if the object is indeed an SHGC, our implementation is to compute t_1 and $scale(t_1)$ separately from each of the image apex tangents, and average them out.

4.4 Experimental Results

Results on a stereo pair of real images of a typical desk lamp are shown in figure 10. The cameras were configured such that the optical axes were parallel with a base line of approximately 25cm long. The lamp was about 75cm away from the cameras. Both cameras have a spatial resolution of 512 by 480 with 8 bits of grey scale. Here we show more details about the intermediate steps used in the hierarchical stereo matching system proposed in [3]. *Edges* are detected from each image using Canny's edge-detector [2], and are linked into edge-contours based on eight-neighbor connectivity. Edge-contours are segmented into *curves* at curvature extremas so that every curve is smooth in itself, and curves are grouped into *contours* based on continuity. *Symmetries* are then detected from each pair of approximately symmetrical contours, and they form *ribbons* if they have proper closures at both ends of the symmetries. The closure at the end of a symmetry can be composed of a curve, a set of multiple curves, or the ends of other symmetries. Very small symmetries are ignored to save computation time. Still a large number of symmetries are left and they form many conflicting ribbons. The ribbons then go through a selection process based on a number of constraints among the ribbons. The selected ribbons and the hierarchies of descriptions in the two images are then used for stereo correspondence. Junctions are extracted from the matched ribbons and labelled as limb-junctions or real junctions from their behavior across the stereo images. Limb edges are also identified during this step. The details of such perceptual grouping and stereo matching processes is given in [3].

We then group neighboring ribbons which share smooth boundaries into objects. Notice that the lamp object is a little bit complex as it consists of two neighboring sections of curved surfaces. Using the Hough Transform method mentioned in section 4.1, we are able to derive the SHGC axes of both curved sections and identify that they both share the similar axis. We then treat the two curved sections as one single SHGC and derive the volumetric descriptions as in the previous example. In figure 10 we overlay the recovered descriptions with the left image to illustrate the performance of our method. Notice that the perspective distortion in the images are significant; the eccentricities of the projected ellipses change gradually along the axis from one end to the other.

5 Conclusion

In this paper we examined the problem of deriving volumetric shape from stereo images. We emphasize that intermediate $2\frac{1}{2}$-D dense or wire-frame descriptions may not be always available from stereo, which is basically why shape from stereo cannot be treated as merely an integration of two modules: depth from stereo, and shape from range data. As a result, the volumetric reconstruction may have to be computed directly from stereo correspondences. We have described how volumetric shape can be reconstructed from stereo by using some shape primitives such as LSHGCs and SHGCs. The methods are based on some invariant properties of the shape models in their 2-D projections. Such properties are not all monocular; we have proposed some properties in

stereo which further help in confirming and reconstructing LSHGCs and SHGCs from stereo images.

Our technique allows dense surface descriptions to be recovered even for objects without any texture at all (texture helps provide dense data in stereo), and it is not restricted to narrow stereo angles or low resolution images. Our technique can also handle objects at close range, which is in fact where stereo is most effective, without being affected by any possible perspective distortion in the projected images. We have shown results for objects with circular cross-sections, but our method is not limited to that. In addition, our method can even allow LSHGC objects with arbitrary cuts across the cones, as well as SHGC objects with oblique cross-sections.

References

[1] T. O. Binford. Visual perception by computer. In *IEEE Conference on Systems and Controls*, Miami, Florida, December 1971.

[2] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[3] C.-K. R. Chung and R. Nevatia. Use of monocular groupings and occlusion analysis in a hierarchical stereo system. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 50–56, Maui, Hawaii, June 1991.

[4] H. S. Lim and T. O. Binford. Curved surface reconstruction using stereo correspondence. In *Proceedings of the DARPA Image Understanding Workshop*, pages 809–819, Cambridge, Massachusetts, April 1988.

[5] R. Mohan and R. Nevatia. Segmentation and description based on perceptual organization. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 333–341, San Diego, California, June 1989.

[6] R. Mohan and R. Nevatia. Using perceptual organization to extract 3-d structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1121–1139, November 1989.

[7] J. Ponce, D. Chelberg, and W. B. Mann. Invariant properties of straight homogeneous generalized cylinders and their contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(9):951–966, September 1989.

[8] K. Rao. *Shape Description from Sparse and Imperfect Data*. PhD thesis, University of Southern California, December 1988. IRIS Technical Report 250.

[9] K. Rao and R. Nevatia. Computing volume descriptions from sparse 3-D data. *International Journal of Computer Vision*, 2(1):33–50, June 1987.

[10] S. A. Shafer and T. Kanade. The theory of straight homogeneous generalized cylinders. Technical Report CMU-CS-083-105, Carnegie-Mellon University, 1983.

[11] F. Ulupinar and R. Nevatia. Recovering shape from contour for shgcs and cgcs. In *Proceedings of the DARPA Image Understanding Workshop*, Pittsburgh, Pennsylvania, September 1990.

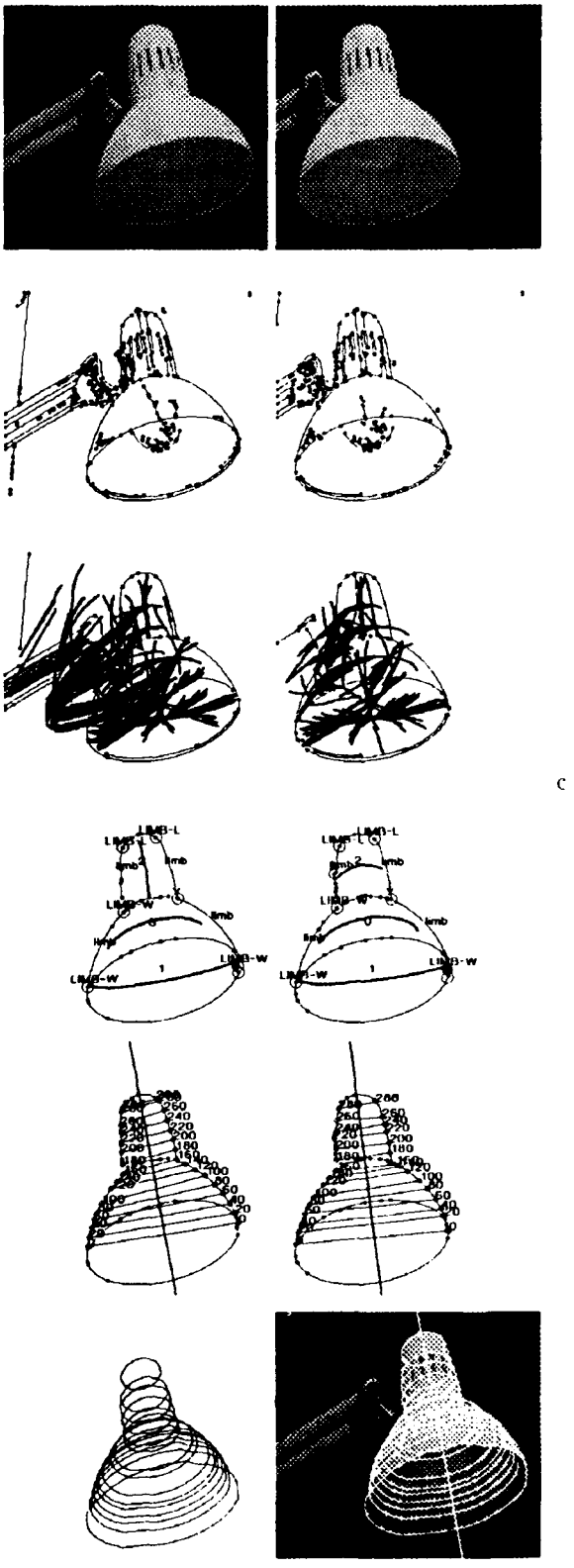


Figure 10: Results of hierarchical stereo matching and volumetric shape recovery for a scene of a lamp (from top to bottom): the left and right images; the left and right edges; the left and right symmetries; the left and right matched ribbons and identified junctions; the left and right extracted image axes and the correspondences across stereo views; the recovered volumetric descriptions projected to and overlaid with the left view.

A Multiple-baseline Stereo Method *

Takeo Kanade **Masatoshi Okutomi†** **Tomoharu Nakahara**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Part I: Theory

M. Okutomi and T. Kanade

Abstract

This paper presents a stereo matching method which uses multiple stereo pairs with various baselines to obtain precise distance estimates without suffering from ambiguity.

In stereo processing, a short baseline means that the estimated distance will be less precise due to narrow triangulation. For more precise distance estimation, a longer baseline is desired. With a longer baseline, however, a larger disparity range must be searched to find a match. As a result, matching is more difficult and there is a greater possibility of a false match. So there is a trade-off between precision and accuracy in matching.

The stereo matching method presented in this paper uses multiple stereo pairs with different baselines generated by a lateral displacement of a camera. Matching is performed simply by computing the sum of squared-difference (SSD) values. The SSD functions for individual stereo pairs are represented with respect to the inverse distance (rather than the disparity, as is usually done), and then are simply added to produce the sum of SSDs. This resulting function is called the SSSD-in-inverse-distance. We show that the SSSD-in-inverse-distance function exhibits a unique and clear minimum at the correct matching position even when the underlying intensity patterns of the scene include ambiguities or repetitive patterns. An advantage of this method is that we can eliminate false matches and increase precision without any search or sequential filtering.

This paper first defines a stereo algorithm based on the SSSD-in-inverse-distance and presents a mathematical analysis to show how the algorithm can remove ambiguity and increase precision. Then, a few experimental results with real stereo images are presented to demonstrate the effectiveness of the algorithm.

1 Introduction

Stereo is a useful technique for obtaining 3-D information from 2-D images in computer vision. In stereo matching, we measure the disparity d, which is the difference between the corresponding points of left and right images. The disparity d is related to the distance z by

$$d = BF\frac{1}{z}, \tag{1}$$

where B and F are baseline and focal length, respectively.

This equation indicates that for the same distance the disparity is proportional to the baseline, or that the baseline length B acts as a magnification factor in measuring d in order to obtain z. That is, the estimated distance is more precise if we set the two cameras farther apart from each other, which means a longer baseline. A longer baseline, however, poses its own problem. Because a longer disparity range must be searched, matching is more difficult and thus there is a greater possibility of a false match. So there is a trade-off between precision and accuracy (correctness) in matching.

One of the most common methods to deal with the problem is a coarse-to-fine control strategy [1] – [5]. Matching is done at a low resolution to reduce false matches and then the result is used to limit the search range of matching at a higher resolution, where more precise disparity measurements are calculated. Using a coarse resolution, however, does not always remove false matches. This is especially true when there is inherent ambiguity in matching, such as a repeated pattern over a large part of the scene (eg., a scene of a picket fence). Another approach to remove false matches and to increase precision is to use multiple images, especially a sequence of densely sampled images along a camera path [6] – [9]. A short baseline between a pair of consecutive images makes the matching or tracking of features easy, while the structure imposed by the camera motion allows integration of the possibly noisy individual measurements into a precise estimate. The integration has been performed either by exploiting constraints on the EPI [6][7] or by a sequential Kalman filtering technique [8][9].

The stereo matching method presented in this paper belongs to the second approach: use of multiple images with different baselines obtained by a lateral displacement of a camera. The matching technique, however, is based on the idea that global mismatches can be reduced by adding the sum of squared-difference (SSD) values from multiple stereo pairs. That is, the SSD values are computed first for each pair of stereo images. We represent the SSD values with respect to the inverse distance $1/z$ (rather than the disparity d, as is usually done). The resulting SSD functions from all stereo pairs are added together to produce the sum of SSDs, which we call SSSD-in-inverse-distance. We show that the SSSD-in-inverse-distance function exhibits a unique and clear minimum at the correct matching position even when the underlying intensity patterns of the scene include ambiguities or repetitive patterns.

There have been stereo techniques that use multiple image pairs taken by cameras which are arranged along a line [10][11][12], in the form of a triangle [13][14][15] (called trinocular stereo), or in the other formation [16]. However, all of these techniques, except [10] and [16], decide candidate points for correspondence in each image pair and then search for the correct combinations of correspondences among them using the geometrical consistencies that they must satisfy. Since the intermediate decisions on correspondences are inherently noisy, ambiguous and multiple, finding the correct combinations requires sophisticated consistency checks and search or filtering. In contrast, our method does not make any

*This research was supported by the Defense Advanced Research Projects Agency (DOD) and monitored by the Avionics Laboratory, Air Force Wright Aeronautical Laboratories, Aeronautical Systems Division (AFSC), Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-87-C-1499, ARPA Order No. 4976, Amendment 20. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of DARPA or the U.S. government.

†Current address: Information Systems Research Center, Canon Inc.890-12 Kashimada, Saiwai-ku, Kawasaki, 211, Japan.

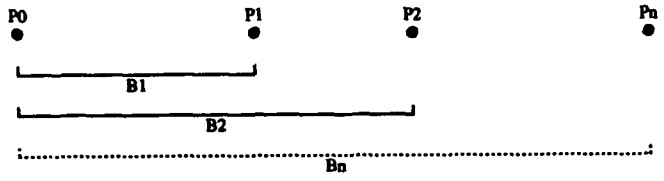


Figure 1: Camera positions for stereo

decisions about the correspondences in each stereo image pair; instead, it simply accumulates the measures of matching (SSDs) from all the stereo pairs into a single evaluation function, ie., SSSD-in-inverse-distance, and then obtains one corresponding point from it. In other words, our method integrates *evidence* for a final decision, rather than filtering intermediate *decisions*. In this sense, Tsai [16] employed strategy very similar to ours: he used multiple images to sharpen the peaks of his overall similarity measures, which he called JMM and WVM. However, the relationship between the improvement of the similarity measures and the camera baseline arrangement was not analyzed, nor was the method tested with real imagery. In this paper we show both mathematical analysis and experimental results with real indoor and outdoor images, which demonstrate how the SSSD-in-inverse-distance function based on multiple image pairs from different baselines can greatly reduce false matches, while improving precision.

In the next section we present the method mathematically and show how ambiguity can be removed and precision increased by the method. Section 3 provides a few experimental results with real stereo images to demonstrate the effectiveness of the algorithm. Section 4 presents conclusions.

2 Mathematical Analysis

The essence of stereo matching is, given a point in one image, to find in another image the corresponding point, such that the two points are the projections of the same physical point in space. This task usually requires some criterion to measure similarity between images. The sum of squared differences (SSD) of the intensity values (or values of preprocessed images, such as bandpass filtered images) over a window is the simplest and most effective criterion. In this section, we define the sum of SSD with respect to the inverse distance (SSSD-in-inverse-distance) for multiple-baseline stereo, and mathematically show its advantage in removing ambiguity and increasing precision. For this analysis, we use 1-D stereo intensity signals, but the extension to two dimensional images is straightforward.

2.1 SSD Function

Suppose that we have cameras at positions $P_0, P_1, \ldots, P_n$ along a line with their optical axes perpendicular to the line and a resulting set of stereo pairs with baselines $B_1, B_2, \ldots, B_n$ as shown in figure 1. Let $f_0(x)$ and $f_i(x)$ be the image pair at the camera positions P_0 and P_i, respectively. Imagine a scene point Z whose distance is z. Its disparity $d_{r(i)}$ for the image pair taken from P_0 and P_i is

$$d_{r(i)} = \frac{B_i F}{z}. \tag{2}$$

We model the image intensity functions $f_0(x)$ and $f_i(x)$ near the matching positions for Z as

$$\begin{aligned} f_0(x) &= f(x) + n_0(x) \\ f_i(x) &= f(x - d_{r(i)}) + n_i(x), \end{aligned} \tag{3}$$

assuming constant distance near Z and independent Gaussian white noise such that

$$n_0(x), n_i(x) \sim N(0, \sigma_n^2). \tag{4}$$

The SSD value $e_{d(i)}$ over a window W at a pixel position x of image $f_0(x)$ for the candidate disparity $d_{(i)}$ is defined as

$$e_{d(i)}(x, d_{(i)}) \equiv \sum_{j \in W} (f_0(x+j) - f_i(x + d_{(i)} + j))^2, \tag{5}$$

where the $\sum_{j \in W}$ means summation over the window. The $d_{(i)}$ that gives a minimum of $e_{d(i)}(x, d_{(i)})$ is determined as the estimate of the disparity at x. Since the SSD measurement $e_{d(i)}(x, d_{(i)})$ is a random variable, we will compute its expected value in order to analyze its behavior:

$$\begin{aligned} & E[e_{d(i)}(x, d_{(i)})] \\ &= E\left[\sum_{j \in W} (f(x+j) - f(x + d_{(i)} - d_{r(i)} + j) \right. \\ &\quad \left. + n_0(x+j) - n_i(x + d_{(i)} + j))^2\right] \\ &= \sum_{j \in W} (f(x+j) - f(x + d_{(i)} - d_{r(i)} + j))^2 + 2N_w\sigma_n^2, \end{aligned} \tag{6}$$

where N_w is the number of the points within the window. For the rest of the paper, $E[]$ denotes the expected value of a random variable. In deriving the above equation, we have assumed that $d_{r(i)}$ is constant over the window. Equation (6) says that naturally the SSD function $e_{d(i)}(x, d_{(i)})$ is *expected* to take a minimum when $d_{(i)} = d_{r(i)}$, i.e., at the right disparity.

Let us examine how the SSD function $e_{d(i)}(x, d_{(i)})$ behaves when there is ambiguity in the underlying intensity function. Suppose that the intensity signal $f(x)$ has the same pattern around pixel positions x and $x + a$,

$$f(x+j) = f(x+a+j), \qquad j \in W \tag{7}$$

where $a \neq 0$ is a constant. Then, from equation (6)

$$E[e_{d(i)}(x, d_{r(i)})] = E[e_{d(i)}(x, d_{r(i)} + a)] = 2N_w\sigma_n^2. \tag{8}$$

This means that ambiguity is expected in matching in terms of positions of minimum SSD values. Moreover, the false match at $d_{r(i)} + a$ appears in exactly the same way for all i; it is separated from the correct match by a for all the stereo pairs. Using multiple baselines does not help to disambiguate.

2.2 SSD with respect to Inverse Distance

Now, let us introduce the *inverse distance* ζ such that

$$\zeta = \frac{1}{z}. \tag{9}$$

>From equation and (2),

$$d_{r(i)} = B_i F \zeta_r \tag{10}$$

$$d_{(i)} = B_i F \zeta, \tag{11}$$

where ζ_r and ζ are the real and the candidate inverse distance, respectively. Substituting equation (11) into (5), we have the SSD with respect to the inverse distance,

$$e_{\zeta(i)}(x, \zeta) \equiv \sum_{j \in W} (f_0(x+j) - f_i(x + B_i F \zeta + j))^2, \tag{12}$$

at position x for a candidate inverse distance ζ. Its expected value is

$$E[e_{\zeta(i)}(x,\zeta)] = \sum_{j\in W}(f(x+j)-f(x+B_iF(\zeta-\zeta_r)+j))^2+2N_w\sigma_n^2. \tag{13}$$

Finally, we define a new evaluation function $e_{\zeta(12\cdots n)}(x,\zeta)$, the sum of SSD functions with respect to the inverse distance (SSSD-in-inverse-distance) for multiple stereo pairs. It is obtained by adding the SSD functions $e_{\zeta(i)}(x,\zeta)$ for individual stereo pairs:

$$e_{\zeta(12\cdots n)}(x,\zeta) = \sum_{i=1}^{n} e_{\zeta(i)}(x,\zeta). \tag{14}$$

Its expected value is

$$\begin{aligned} E[e_{\zeta(12\cdots n)}(x,\zeta)] &= \sum_{i=1}^{n} E[e_{\zeta(i)}(x,\zeta)] \\ &= \sum_{i=1}^{n}\sum_{j\in W}(f(x+j)-f(x+B_iF(\zeta-\zeta_r)+j))^2 \\ &\quad +2nN_w\sigma_n^2. \end{aligned} \tag{15}$$

In the next three subsections, we will analyze the characteristics of these evaluation functions to see how ambiguity is removed and precision is improved.

2.3 Elimination of Ambiguity (1)

As before, suppose the underlying intensity pattern $f(x)$ has the same pattern around x and $x+a$ (equation (7)). Then, according to equation (13), we have

$$E[e_{\zeta(i)}(x,\zeta_r)] = E[e_{\zeta(i)}(x,\zeta_r+\frac{a}{B_iF})] = 2N_w\sigma_n^2. \tag{16}$$

We still have an ambiguity; a minimum is expected at a false inverse distance $\zeta_f = \zeta_r + \frac{a}{B_iF}$. However, an important point to be observed here is that this minimum for the false inverse distance ζ_f changes its position as the baseline B_i changes, while the minimum for the correct inverse distance ζ_r does not. This is the property that the new evaluation function, the SSSD-in-inverse-distance (14), exploits to eliminate the ambiguity. For example, suppose we use two baselines B_1 and B_2 ($B_1 \neq B_2$). >From equation (15)

$$\begin{aligned} &E[e_{\zeta(12)}(x,\zeta)] \\ &= \sum_{j\in W}(f(x+j)-f(x+B_1F(\zeta-\zeta_r)+j))^2 \\ &+ \sum_{j\in W}(f(x+j)-f(x+B_2F(\zeta-\zeta_r)+j))^2 \\ &+ 4N_w\sigma_n^2. \end{aligned} \tag{17}$$

We can prove that

$$E[e_{\zeta(12)}(x,\zeta)] > 4N_w\sigma_n^2 = E[e_{\zeta(12)}(x,\zeta_r)] \quad \text{for } \zeta \neq \zeta_r. \tag{18}$$

(refer to appendix A) In words, $e_{\zeta(12)}(x,\zeta)$ is *expected* to have the smallest value at the correct ζ_r. That is, the ambiguity is likely to be eliminated by use of the new evaluation function with two different baselines.

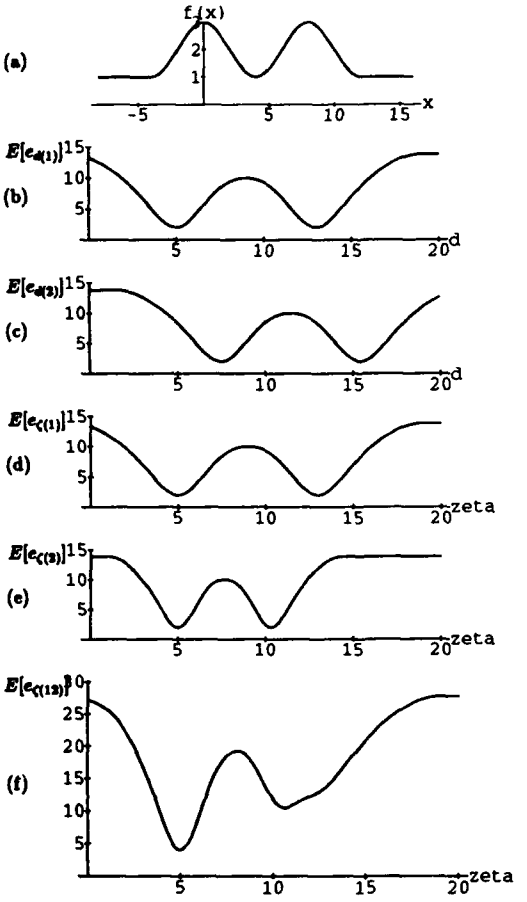


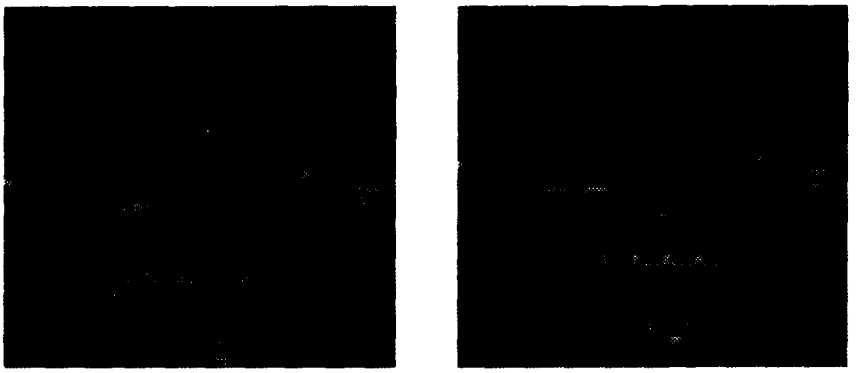
Figure 2: Expected values of evaluation functions: (a) Underlying function; (b) $E[e_{d(1)}]$; (c) $E[e_{d(2)}]$; (d) $E[e_{\zeta(1)}]$; (e) $E[e_{\zeta(2)}]$; (f) $E[e_{\zeta(12)}]$

We can illustrate this using synthesized data. Suppose the point whose distance we want to determine is at $x = 0$ and the underlying function $f(x)$ is given by

$$f(x) = \begin{cases} cos(\frac{\pi}{4}x)+2 & \text{if } -4 < x < 12 \\ 1 & \text{if } x \leq -4 \text{ or } 12 \leq x. \end{cases} \tag{19}$$

Figure 2 (a) shows a plot of $f(x)$. Assuming that $d_{r(1)} = 5$, $\sigma_n^2 = 0.2$, and the window size is 5, the expected values of the SSD function $e_{d(1)}(x, d_{(1)})$ are as shown in figure 2 (b). We see that there is an ambiguity: the minima occur at the correct match $d_{(1)} = 5$ and at the false match $d_{(1)} = 13$. Which match will be selected will depend on the noise, search range, and search strategy. Now suppose we have a longer baseline B_2 such that $\frac{B_2}{B_1} = 1.5$. >From equations (6) and (10), we obtain $E[e_{d(2)}]$ as shown in figure 2 (c). Again we encounter an ambiguity, and the separation of the two minima is the same.

Now let us evaluate the SSD values with respect to the inverse distance ζ rather than the disparity d by using equations (12) through (15). The expected values of the SSD measurements $E[e_{\zeta(1)}]$ and $E[e_{\zeta(2)}]$ with baselines B_1 and B_2 are shown in figures 2 (d) and (e), respectively (the plot is normalized such that $B_1F = 1$). Note that the minima at the correct inverse distance ($\zeta = 5$) does not move, while the minima for the false



(a) (b)

Figure 3: "Town" data set: (a) Image0; (b) Image9

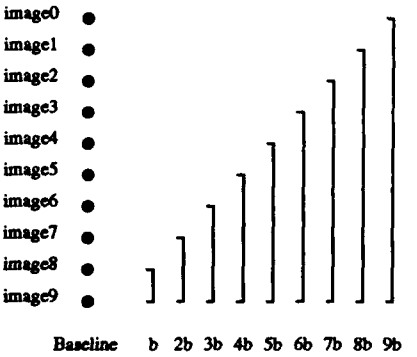


Figure 4: "Town" data set image sequence

match changes its position as the baseline changes. When the two functions are added to produce the SSSD-in-inverse-distance, its expected values $E[e_{\zeta(12)}]$ are as shown in figure 2 (f). We can see that the ambiguity has been reduced because the SSSD-in-inverse-distance has a smaller value at the correct match position than at the false match.

2.4 Elimination of Ambiguity (2)

An extreme case of ambiguity occurs when the underlying function $f(x)$ is a periodic function, like a scene of a picket fence. We can show that this ambiguity can also be eliminated.

Let $f(x)$ be a periodic function with period T. Then, each $e_{\zeta(i)}(x,\zeta)$ is expected to be a periodic function of ζ with the period $\frac{T}{B_iF}$. This means that there will be multiple minima of $e_{\zeta(i)}(x,\zeta)$ (i.e., ambiguity in matching) at intervals of $\frac{T}{B_iF}$ in ζ. When we use two baselines and add their SSD values, the resulting $e_{\zeta(12)}(x,\zeta)$ will be still a periodic function of ζ, but its period T_{12} is increased to

$$T_{12} = LCM\left(\frac{T}{B_1F}, \frac{T}{B_2F}\right), \qquad (20)$$

where $LCM()$ denotes Least Common Multiple. That is, the period of the expected value of the new evaluation function can be made longer than that of the individual stereo pairs. Furthermore, it can be controlled by choosing the baselines B_1 and B_2 appropriately so that the expected value of the evaluation function has only one minimum within the search range. This means that using multiple-baseline stereo pairs simultaneously can eliminate ambiguity, although each individual baseline stereo may suffer from ambiguity.

We illustrate this by using real stereo images. Figure 3(a) shows an image of a sample scene. At the top of the scene there is a grid board whose intensity function is nearly periodic. We took ten images of this scene by shifting the camera vertically as in figure 4. The actual distance between consecutive camera positions is 0.05 inches. Let this distance be b. Figure 3 shows the first and the last images of the sequence. We selected a point x within the repetitive grid board area in image9. The SSD values $e_{\zeta(i)}(x,\zeta)$ over 5-by-5-pixel windows are plotted for various baseline stereo pairs in figure 5. The horizontal axis of all the plots is the inverse distance, normalized such that $8bF = 1$. Figure 5 illustrates the trade-off between precision and ambiguity in terms of baselines. That is, for a shorter baseline, there are fewer minima (i.e. less ambiguity), but the SSD curve is flatter (i.e. less precise localization). On the other hand, for a longer baseline, there are more minima (i.e. more ambiguity), but the curve near the minimum is sharper; that is, the estimated distance is more precise if we can find the correct one.

Now, let us take two stereo image pairs: one with $B = 5b$ and the other with $B = 8b$. In figure 6, the dashed curve and the dotted curve show the SSD for $B = 5b$ and $B = 8b$, respectively. Let us suppose the search range goes from 0 to 20 in the horizontal axis, which in this case corresponds to 12 to ∞ inches in distance. Though the SSD values take a minimum at the correct answer near $\zeta = 5$, there are also other minima for both cases. The solid curve shows the evaluation function for the multiple-baseline stereo, which is the sum of the dashed curve and the dotted curve. The solid curve shows only one clear minimum; that is, the ambiguity is resolved.

So far, we have considered using only two stereo pairs. We can easily extend the idea to multiple-baseline stereo which uses more than two stereo pairs. Corresponding to equation (20), the period of $E[e_{\zeta(12\cdots n)}(x,\zeta)]$ becomes

$$T_{12,\ldots,n} = LCM\left(\frac{T}{B_1F}, \frac{T}{B_2F}, \ldots, \frac{T}{B_nF}\right) \qquad (21)$$

where $B_1, B_2, \ldots, B_n$ are baselines for each stereo pair.

We will demonstrate how the ambiguity can be further reduced by increasing the number of stereo pairs. >From the data of figure 4, we first choose image1 and image9 as a long baseline stereo pair, ie. (1) $B = 8b$. Then, we increase the number of stereo pairs by dividing the baseline between image1 and image9, i.e. (2) $B = 4b$ and $8b$, (3) $B = 2b$, $4b$, $6b$ and $8b$, (4) $B = b$, $2b$, $3b$, $4b$, $5b$, $6b$, $7b$ and $8b$. Figure 7 demonstrates that the SSSDs-in-inverse-distance shows the minimum at the correct position more clearly as more stereo pairs are used.

2.5 Precision

We have shown that ambiguities can be resolved by using the SSSD-in-inverse-distance computed from multiple baseline stereo pairs. The technique also increases precision in estimating the true inverse distance. We can show this by analyzing the statistical characteristics of the evaluation functions near the correct match.

>From equations (3), (10), and (12), we have

$$e_{\zeta(i)}(x,\zeta) = \sum_{j\in W}(f(x+j) - f(x+B_iF(\zeta-\zeta_r)+j) + n_0(x+j) - n_i(x+B_iF\zeta+j))^2. \qquad (22)$$

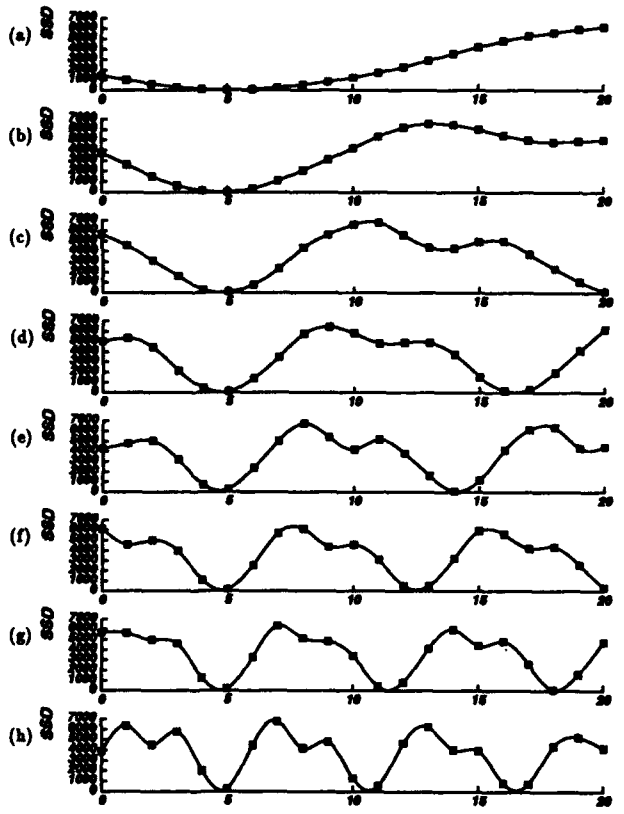


Figure 5: SSD values vs. inverse depth: (a) $B = b$; (b) $B = 2b$; (c) $B = 3b$; (d) $B = 4b$; (e) $B = 5b$; (f) $B = 6b$; (g) $B = 7b$; (h) $B = 8b$. The horizontal axis is normalized such that $8bF = 1$.

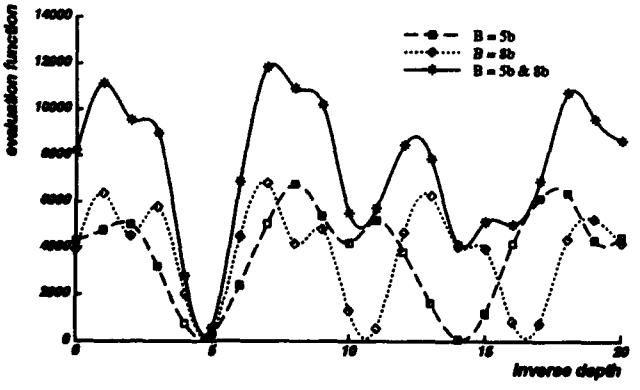


Figure 6: Combining two stereo pairs with different baselines

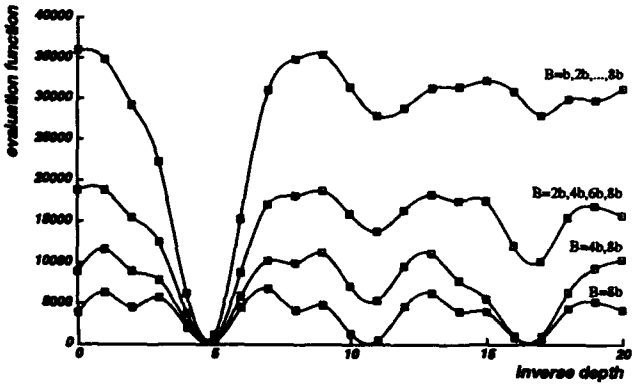


Figure 7: Combining multiple baseline stereo pairs

By taking the Taylor expansion about $\zeta = \zeta_r$ up to the linear terms, we obtain

$$f(x+B_iF(\zeta-\zeta_r)+j) \approx f(x+j)+B_iF(\zeta-\zeta_r)f'(x+j). \quad (23)$$

Substituting this into equation (22), we can approximate $e_{\zeta(i)}(x,\zeta)$ near ζ_r by a quadratic form of ζ:

$$\begin{aligned} & e_{\zeta(i)}(x,\zeta) \\ \approx & \sum_{j\in W}(-B_iF(\zeta-\zeta_r)f'(x+j) \\ & +n_0(x+j)-n_i(x+B_iF\zeta+j))^2 \\ = & B_i^2F^2a(x)(\zeta-\zeta_r)^2+2B_iFb_i(x)(\zeta-\zeta_r)+c_i(x), \end{aligned} \quad (24)$$

where

$$a(x) = \sum_{j\in W}(f'(x+j))^2 \quad (25)$$

$$b_i(x) = \sum_{j\in W}f'(x+j)(n_i(x+B_iF\zeta+j)-n_0(x+j)) \quad (26)$$

$$c_i(x) = \sum_{j\in W}(n_i(x+B_iF\zeta+j)-n_0(x+j))^2. \quad (27)$$

The estimated inverse distance $\xi_{r(i)}$ is the value ζ that makes equation (24) minimum;

$$\xi_{r(i)} = \zeta_r - \frac{b_i(x)}{B_iFa(x)}. \quad (28)$$

Since $E[b_i(x)] = 0$, the expected value of the estimate $\xi_{r(i)}$ is the correct value ζ_r, but it varies due to the noise. The variance of this estimate is:

$$\begin{aligned} Var(\xi_{r(i)}) &= \frac{Var(b_i(x))}{B_i^2F^2(a(x))^2} \\ &= \frac{2\sigma_n^2}{B_i^2F^2a(x)}. \end{aligned} \quad (29)$$

Basically, this equation states that for the same amount of image noise σ_n^2, the variance is smaller (the estimate is more precise) as the baseline B_i is longer, or as the variation of intensity signal, $a(x)$, is larger.

We can follow the same analysis for $e_{\zeta(12\cdots n)}(x,\zeta)$ of (14), the new evaluation function with multiple baselines. Near ζ_r, it is

$$e_{\zeta(12\cdots n)}(x,\zeta) \approx \left(\sum_{i=1}^{n} B_i^2\right) F^2 a(x)(\zeta-\zeta_r)^2 + 2F\left(\sum_{i=1}^{n} B_i b_i(x)\right)(\zeta-\zeta_r) + \sum_{i=1}^{n} c_i(x). \quad (30)$$

The variance of the estimated inverse distance $\xi_{r(12\cdots n)}$ that minimizes this function is

$$Var(\xi_{r(12\cdots n)}) = \frac{2\sigma_n^2}{\left(\sum_{i=1}^{n} B_i^2\right) F^2 a(x)}. \quad (31)$$

>From equations (29) and (31), we see that

$$\frac{1}{Var(\xi_{r(12\cdots n)})} = \sum_{i=1}^{n} \frac{1}{Var(\xi_{r(i)})}. \quad (32)$$

The inverse of the variance represents the precision of the estimate. Therefore, equation (32) means that by using the SSSD-in-inverse-distance with multiple baseline stereo pairs, the estimate becomes more precise. We can confirm this characteristic in figures 6 and 7 by observing that the curve around the correct inverse distance becomes sharper as more baselines are used.

3 Experimental Results

This section presents experimental results of the multiple-baseline stereo based on SSSD-in-inverse-distance with real 2D images. A complete description of the algorithm is included in Appendix B.

The first result is for the "Town" data set that we showed in figure 3. Figures 8 (a) and (b) are the distance map and its isometric plot with a short baseline, $B = 3b$. The result with a single long baseline, $B = 9b$, is shown in figure 9. Comparing these two results, we observe that the distance map computed by using the long baseline is smoother on flat surfaces, i.e., more precise, but has gross errors in matching at the top of the scene because of the repeated pattern. These results illustrate the trade-off between ambiguity and precision. Figure 10, on the other hand, shows the distance map and its isometric plot obtained by the new algorithm using three different baselines, $3b$, $6b$, and $9b$. For comparison, the corresponding oblique view of the scene is shown in figure 11. We can note that the computed distance map is less ambiguous *and* more precise than those of the single-baseline stereo.

Figure 12 shows another data set used for our experiment. Figures 13 and 14 compare the distance maps computed from the short baseline stereo and the long baseline stereo: the longer baseline is five times longer than the short one. For comparison, the actual oblique view roughly corresponding to the isometric plot is shown in figure 15. Though no repetitive patterns are apparent in the images, we can still observe gross errors in the distance map obtained with the long baseline due to false matching. In contrast, the result from the multiple-baseline stereo shown in figure 16 demonstrates both the advantage of unambiguous matching with a short baseline and that of precise matching with a long baseline.

4 Conclusions

In this paper, we have presented a new stereo matching method which uses multiple baseline stereo pairs. This method can overcome the trade-off between precision and accuracy (avoidance of false matches) in stereo. The method is rather straightforward: we represent the SSD values for individual stereo pairs as a function of the inverse distance, and add those functions. The resulting function, the SSSD-in-inverse-distance, exhibits an unambiguous and sharper minimum at the correct matching position. As a result there is no need for search or sequential estimation procedures.

The key idea of the method is to relate SSD values to the inverse distance rather than the disparity. As an afterthought, this idea is natural. Whereas disparity is a function of the baseline, there is only one true (inverse) distance for each pixel position for all of the stereo pairs. Therefore there must be a single minimum for the SSD values when they are summed and plotted with respect to the inverse distance. We have shown the advantage of the proposed method in removing ambiguity and improving precision by analytical and experimental results.

Acknowledgment

The authors would like to thank John Krumm for his useful comments on this paper. Keith Gremban, Jim Rehg and Carol Novak have read the manuscript and improved its readability substantially.

A SSSD-in-inverse-distance for Ambiguous Pattern

Proposition: Suppose that there are two and only two repetitions of the same pattern around positions x and $x+a$ where $a \neq 0$ is a constant. That is, for $j \in W$

$$f(x+j) = f(\xi+j), \qquad \text{if and only if } \xi = x \text{ or } \xi = x+a. \quad (33)$$

Then, if $B_1 \neq B_2$, for $\forall\zeta, \zeta \neq \zeta_r$,

$$\begin{aligned} & E[e_{\zeta(12)}(x,\zeta)] \\ = & \sum_{j\in W}(f(x+j) - f(x+B_1F(\zeta-\zeta_r)+j))^2 \\ + & \sum_{j\in W}(f(x+j) - f(x+B_2F(\zeta-\zeta_r)+j))^2 + 4N_w\sigma_n^2 \\ > & \; 4N_w\sigma_n^2 = E[e_{\zeta(12)}(x,\zeta_r)]. \end{aligned} \quad (34)$$

Proof: Tentatively suppose that for $\exists\zeta_f, \zeta_f \neq \zeta_r$,

$$\begin{aligned} & \sum_{j\in W}(f(x+j) - f(x+B_1F(\zeta_f-\zeta_r)+j))^2 \\ + & \sum_{j\in W}(f(x+j) - f(x+B_2F(\zeta_f-\zeta_r)+j))^2 \\ = & \; 0. \end{aligned} \quad (35)$$

Then, it must be the case that

$$\begin{aligned} f(x+j) &= f(x+a_1+j) \\ \text{and} \quad f(x+j) &= f(x+a_2+j), \end{aligned} \quad (36)$$

for $j \in W$, where

$$\begin{aligned} a_1 &= B_1F(\zeta_f-\zeta_r) \\ a_2 &= B_2F(\zeta_f-\zeta_r). \end{aligned}$$

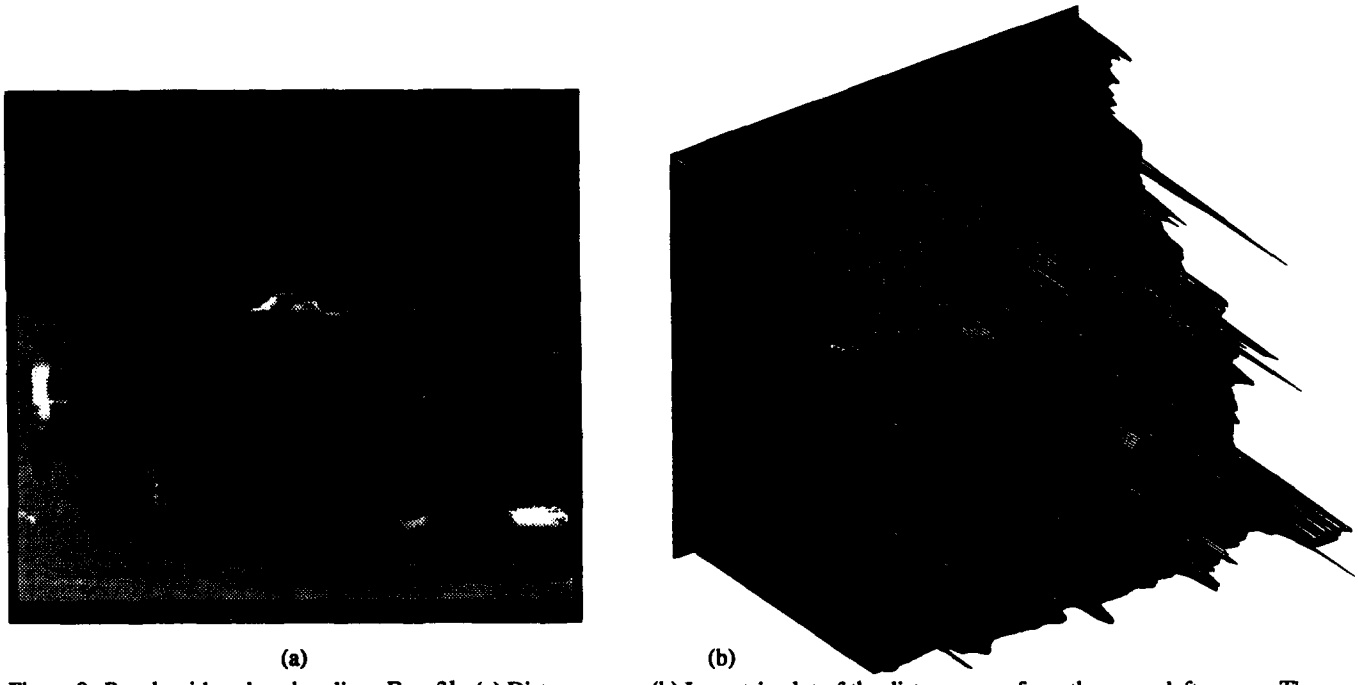
Since $B_1 \neq B_2$ and $\zeta_r \neq \zeta_f$,

$$a_1 \neq a_2. \quad (37)$$

So, we have

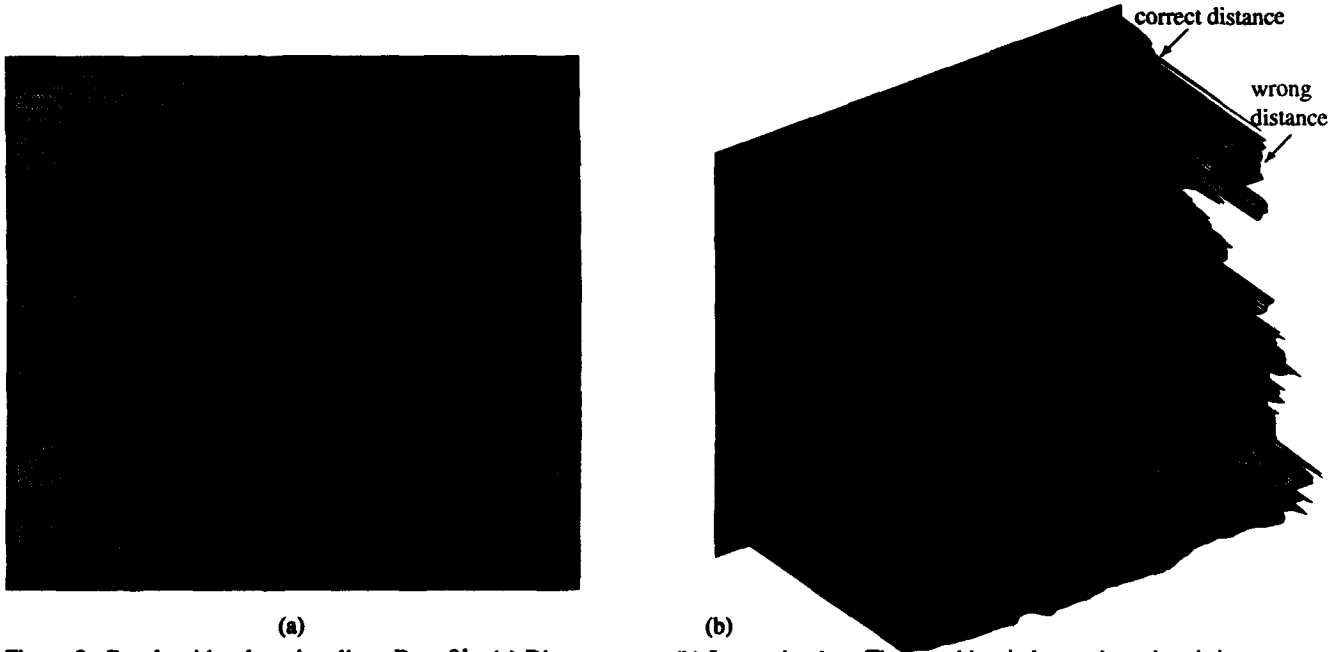
$$f(x+j) = f(\xi+j), \qquad \text{for } \xi = x, x+a_1, \text{ or } x+a_2. \quad (38)$$

Since this contradicts assumption (33), equation (35) does not hold. Its left hand side must be positive. Hence (34) holds.



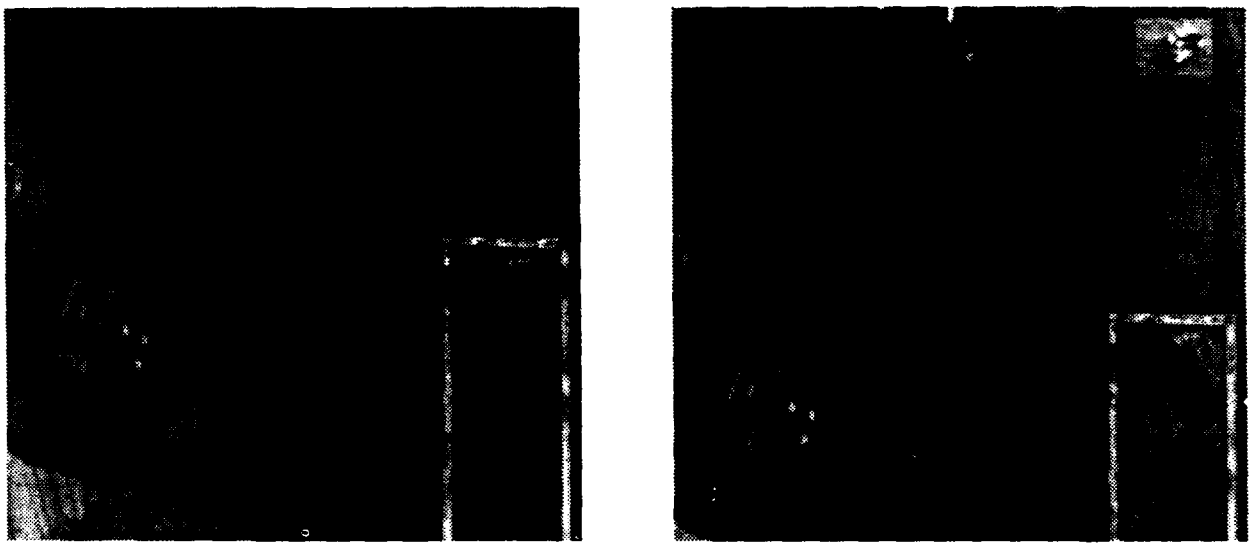
(a) (b)

Figure 8: Result with a short baseline, $B = 3b$: (a) Distance map; (b) Isometric plot of the distance map from the upper left corner. The matching is mostly correct, but very noisy.



(a) (b)

Figure 9: Result with a long baseline, $B = 9b$: (a) Distance map; (b) Isometric plot. The matching is less noisy when it is correct. However, there are many gross mistakes, especially in the top of the image where, due to a repetitive pattern, the matching is completely wrong.



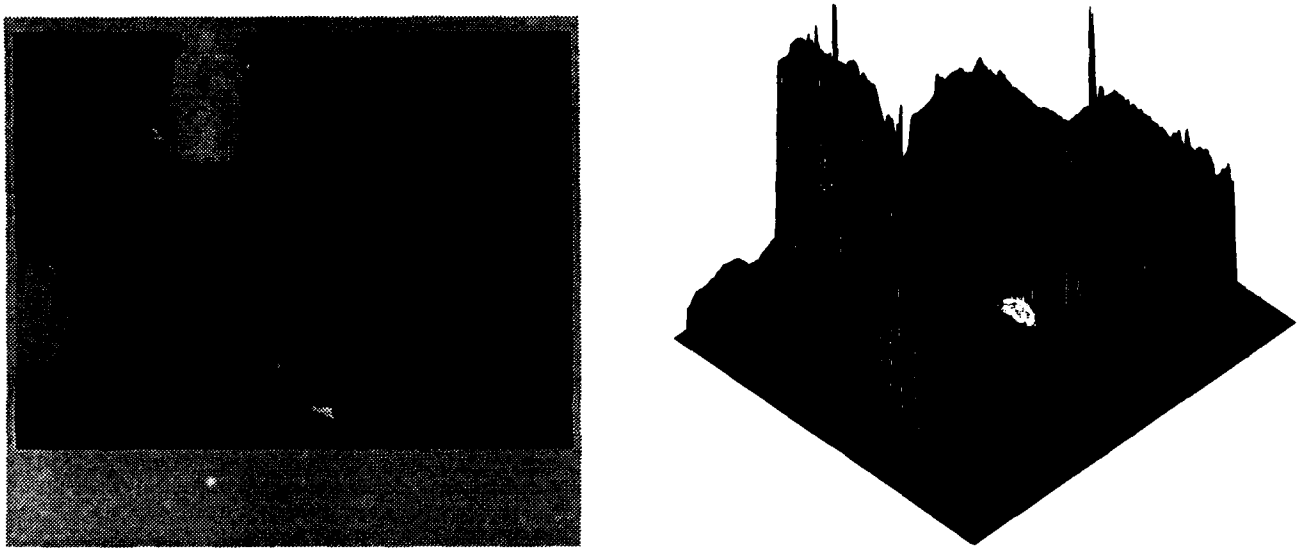
(a) (b)

Figure 12: "Coal mine" data set, long-baseline pair



(a) (b)

Figure 13: Result with a short baseline: (a) Distance map; (b) Isometric plot of the distance map viewed from the lower left corner



(a) (b)

Figure 14: Result with a long baseline: (a) Distance map; (b) Isometric plot

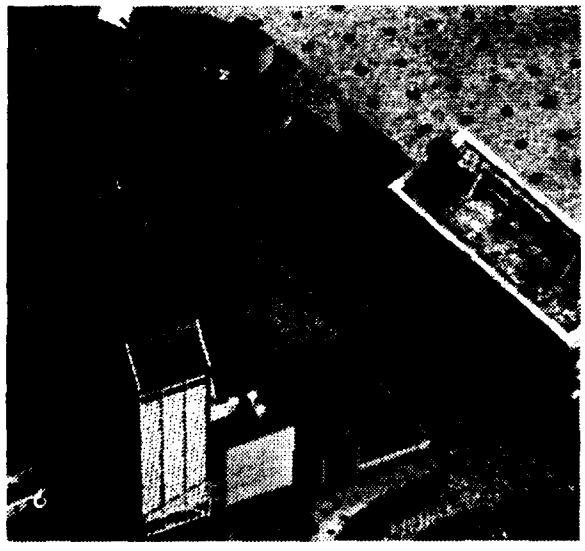
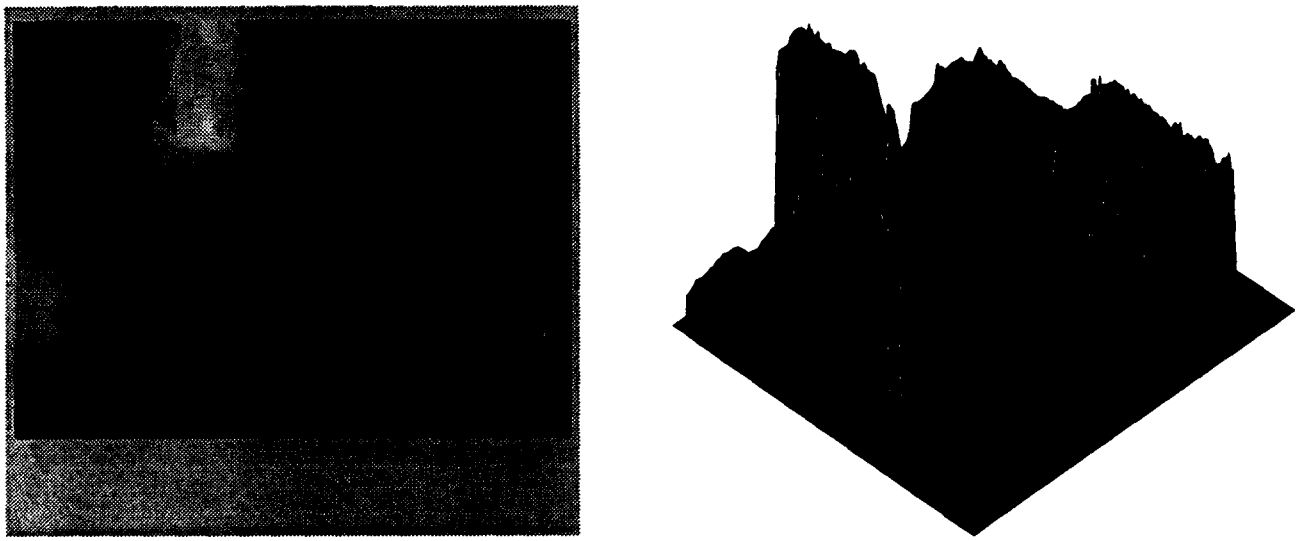


Figure 15: Oblique view



(a) (b)

Figure 16: Multiple baselines: (a) Distance map; (b) Isometric plot

B Multiple-Baseline Stereo Algorithm

We present a complete description of the stereo algorithm using multiple-baseline stereo pairs. The task is, given n stereo pairs, find the ζ that minimizes the SSSD-in-inverse-distance function,

$$SSSD(x,\zeta) = \sum_{i=1}^{n} \sum_{j \in W} (f_0(x+j) - f_i(x + B_i F \zeta + j))^2. \quad (39)$$

We will perform this task in two steps: one at pixel resolution by minimum detection and the other at sub-pixel resolution by iterative estimation.

Minimum of SSSD at Pixel Resolution

For convenience, instead of using the inverse distance, we normalize the disparity values of individual stereo pairs with different baselines to the corresponding values for the largest baseline. Suppose $B_1 < B_2 < \cdots < B_n$. We define the baseline ratio R_i such that

$$R_i = \frac{B_i}{B_n}. \quad (40)$$

Then,

$$B_i F \zeta = R_i B_n F \zeta = R_i d_{(n)}, \quad (41)$$

where $d_{(n)}$ is the disparity for the stereo pair with baseline B_n. Substituting this into equation (39),

$$SSSD(x, d_{(n)}) = \sum_{i=1}^{n} \sum_{j \in W} (f_0(x+j) - f_i(x + R_i d_{(n)} + j))^2. \quad (42)$$

We compute the SSSD function for a range of disparity values at the pixel resolution, and identify the disparity that gives the minimum. Note that pixel resolution for the image pair with the longest baseline (B_n) requires calculation of SSD values at sub-pixel resolution for other shorter baseline stereo pairs.

Iterative Estimation at Sub-pixel Resolution

Once we obtain disparity at pixel resolution for the longest baseline stereo, we improve the disparity estimate to sub-pixel resolution by an iterative algorithm presented in [12][17]. For this iterative estimation, we use only the image pair $f_0(x)$ and $f_n(x)$ with the longest baseline. This is due to a few reasons. First, since the pixel-level estimate was obtained by using the SSSD-in-inverse-distance, the ambiguity has been eliminated and only improvement of precision is intended at this stage. Second, using only the longest-baseline image pair reduces the computational requirement for SSD calculation by a factor of n, and yet does not degrade precision too significantly.

In the experiments shown in section 3, we used the following algorithm for sub-pixel estimation: Let $d_{0(n)}$ be the initial disparity estimate obtained at pixel resolution. Then, a more precise estimate is computed by calculating the following two quantities:

$$\Delta d_{(n)} = \frac{\sum_{j \in W} (f_0(x+j) - f_n(x + d_{0(n)} + j)) f_n'(x + d_{0(n)} + j)}{\sum_{j \in W} (f_n'(x + d_{0(n)} + j))^2} \quad (43)$$

$$\sigma^2_{\Delta d_{(n)}} = \frac{2\sigma_n^2}{\sum_{j \in W} (f_n'(x + d_{0(n)} + j))^2}, \quad (44)$$

The value $\Delta d_{(n)}$ is the estimate of the correction of the disparity to further minimize the SSD, and $\sigma^2_{\Delta d_{(n)}}$ is its variance. We iterate this procedure by replacing $d_{0(n)}$ by

$$d_{0(n)} \leftarrow d_{0(n)} + \Delta d_{(n)} \quad (45)$$

until the estimate converges or up to a certain maximum number of iterations.

References

[1] D. Marr and T. Poggio. A theory of human stereo vision. In *Proc. Roy. Soc. London*, volume vol. B 204, pages 301–328, 1979.

[2] W. E. L.Grimson. Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(1): 17-34, Jan. 1985.

[3] Stephen T. Barnard. Stochastic stereo matching over scale. *International Journal of Computer Vision*, pages 17-32, 1989.

[4] M. J. Hannah. A system for digital stereo image matching. *Photogrammetric Engineering and Remote Sensing*, 55(12): 1765-1770, Dec. 1989

[5] Jer-sen Chen and Gerard Medioni. Parallel multiscale stereo matching using adaptive smoothing. in *ECCV'90*, pages99-103, 1990.

[6] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: An Approach to determining structure from motion. *International Journal of Computer Vision*, 1(1), 1987.

[7] Masanobu Yamamoto. The image sequence analysis of three-dimensional dynamic scenes. Technical Report 893, Electrotechnical Laboratory - Agency of Industrial Science and Technology, Tsukuba, Ibaraki, Japan, May 1988.

[8] Larry Matthies, Richard Szeliski, and Takeo Kanade. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3:209-236, 1989.

[9] Joachim Heel. Dynamic motion vision. In *Proceedings of the DARPA Image Undestanding Workshop*, pages 702-713, Palo Alto, Ca, May 23-26 1989.

[10] B. Wilcox. Telerobotics and Mars Rover research at JPL. Lecture at CMU, Oct. 1987.

[11] Hans P. Moravec. Visual Mapping by a robot rover. In *Proc. IJCAI'79*, pages 598-600, 1079.

[12] Larry Matthies and Masatoshi Okutomi. A bayesian foundation for active stereo vision. In *SPIE, Sensor Fusion II: Human and Machine Strategies*, pages 62-74, Nov. 1989.

[13] M. Yachida, Y. Kitamura, and M. Kimachi. Trinocularvision: New approach for correspondence problem. In *Proc. ICPR*, pages 1041-1044, 1986.

[14] Victor J. Milenkovic and Takeo Kanade. Trinocular vision using photometric and edge orientation constraints. In *Proceedings of the Image Understanding Workshop*, pages 163-175, Miami Beach, Florida, Dec. 1985.

[15] N. Ayache and F. Lustman. Fast and reliable passive trinocular stereo vision. In *Proc. ICCV'87*, pages 422-426, 1987.

[16] Roger Y. Tsai. Multiframe image point matching and 3-d surface reconstraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(No.2), March 1983.

[17] Masatoshi Okutomi and Takeo Kanade. A locally adaptive window for signal matching. In *Proc. of Int'l Conf. on Computer Vision*, Dec. 1990.

Part II: Experiments on Outdoor Scenes

T. Nakahara and T. Kanade

In "Part I: Theory," we explained how multiple stereo pairs with various baselines were used to obtain precise depth estimates without suffering from ambiguity. The algorithm was tested with indoor images which were taken under well controlled conditions in the Calibrated Imaging Laboratory.

This algorithm is applied to outdoor scenes including variable lighting conditions and large depth range. While Okutomi and Kanade used stereo pairs acquired by moving a camera horizontally, we use stereo pairs taken by moving a camera in both horizontal and vertical orientations. Taking stereo images with two orthogonal baseline orientations removes ambiguity and increases precision without suffering from the orientation of the features in a scene. And we also show that the shapes of the sum of squared-difference (SSD) values near the estimate may indicate the reliability of the match, and suggest a method to classify matches into various types, such as good matches and mismatches with occlusion or sparse features.

1. Horizontal baselines Experiment

The experimental setup for acquiring stereo pairs is illustrated in fig. 1. The images are acquired by moving a camera horizontally. The distance between adjacent camera positions is constant. Table 1 describes the image acquisition parameters. Typically, the distance from the camera to the nearest object is 19 m and the baseline length ranges from 19.05 mm for the closest camera pair to 114.1 mm for the farthest.

As illustrated in fig. 2, first the input images are preprocessed with Laplacian of Gaussian (LOG) filter to reduce photometric distortion. A 5x5 window is used for Gaussian and a 3x3 window is used for Laplacian. Then the multiple-baseline stereo is used to compute the inverse depth with a 9x9 window for SSD computation. Typically, the number of the stereo pairs is 6, the image size is 240x256, and the total disparity range is 9 pixels, as summarized in table 2.

1.1. Results

We experiment with three data sets, "Shrubbery," "Parking meters," and "Sand" for the horizontal experiments.

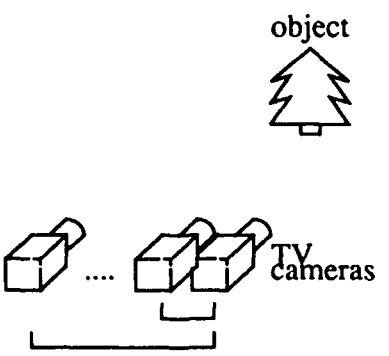


Fig. 1 Setup for horizontal baselines

1.1.1. Shrubbery

Fig. 4 shows the "Shrubbery" data set which consists of six stereo pairs. The maximum disparity between the adjacent images is around two pixels. Fig. 5 is a LOG preprocessing result of one of the images. Fig. 6 is the isometric plot of the resultant depth map. We observe that the shrubberies at the left and in the center are well separated, and the depth jump around the sign board and the top of the signpost are clearly distinct from the wall. We can see a round shrubbery at the right and some pebbles on the road. Some mismatches are observed at the curb, because the features in this area are almost parallel to the epipolar line.

1.1.2. Parking meters

This data set includes seven stereo pairs. Fig. 8 is the isometric plot of the depth map. The following portions in the scene are well estimated: the three parking meters in front of the shrubberies, the side view of the sign board which is between the second and the third parking meters, and the large depth gap between the front and the back building. There are some mismatches at the back door of the car because of sparse features in this part.

1.1.3. Sand

This scene contains natural rough surfaces like sand and a rock as shown in fig. 9. Five stereo pairs are used for this data set. Fig. 10 is the isometric plot of the depth map. We observe that the two rocks and the sand are well estimated. Many mismatches, however, occur at the border between the black wall and the white curtain. The features in this portion are parallel to the epipolar line and are low in density.

1.2. Shapes of SSD and SSSD Curves

In this section we show that the shapes of the SSDs-in-inverse-depth may indicate the reliability of a match and suggest the cause of a mismatch. For this purpose, we examine the shapes of the SSD and the sum of the SSD (SSSD) in three typical cases: a good match, a mismatch with occlusion, and a mismatch with sparse features.

First, we examine the shapes of the SSD and the SSSD for a point whose depth is precisely and accurately estimated, such as a point ① on the sand in fig. 9. Fig. 11 plots 12 curves of individual SSDs and the resultant SSSD for this point. We observe that the minimum of the SSD of each baseline takes place at the same position and the curvature of the SSD near the minimum of the SSSD becomes sharper as the baseline becomes longer. The SSSD exhibits a unique and clear minimum at the correct matching position.

Let us approximate individual SSD's curves by a quadratic equation near the minimum position. From equations (22) - (29) in "Part I: Theory," we expect the following:

- The inverse depth at which the SSD values take the minimum is expected to be the same over the various baselines.
- The curvature is proportional to the square of the baseline length.
- Variance of differences between the inverse depth at the minimum position of each SSD and the final estimated inverse depth is inversely proportional to the square of the baseline length.

Fig. 12 (a), (b), and (c) show the above-mentioned theoretically expected values and experimental measurements for the case of a good match shown in fig. 11. The measurements coincide well with the theoretical values.

Second, we look into the occlusion case, such as a point ② at the right of the first parking meter head in fig. 7. The correspondence points exist in shorter baselines. As the baseline becomes longer, occlusion, however, occurs and matching is not possible. The SSD and the SSSD for the point ② are shown in fig. 13 (a). The inverse depth at the minimum of the SSD of each baseline gradually shifts from the true position to a false position. The SSSD does not show a clear minimum. As shown in fig.13 (b), (c), and (d), the theoretically expected values and the measurements coincide where the baselines are short but differ greatly where the baselines are long.

The third case is a point with sparse features like a point ③ at the black wall in fig. 9. As shown in fig. 13 (e), the SSD curve of each baseline is almost flat over the inverse depth range with no obvious minimum. Consequently the SSSD does not have the minimum.

Another observation for the part of a depth map with mismatch or noisy measurements is the problem of the orientation of the features in a scene. We can not obtain good depth estimates near the curb portion in "Shrubbery" or the border between the black

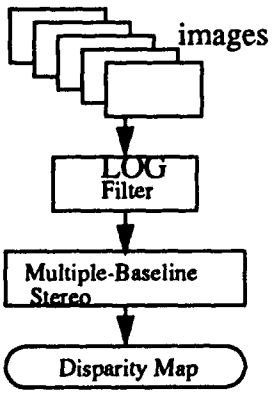


Fig. 2 Procedure

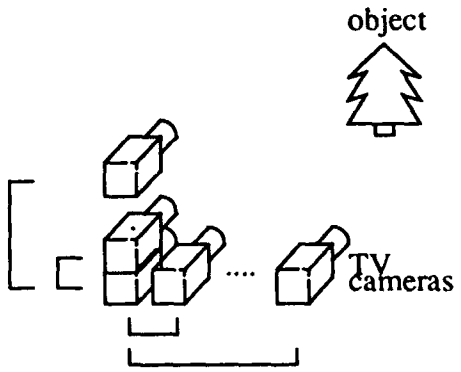


Fig. 3 Setup for horizontal and vertical baselines

wall and the curtain in "Sand," because the image contains only horizontal features. The solution is to use additional stereo image pairs taken by cameras aligned in vertical direction. Combining the information of the vertical baseline with the information of the horizontal baseline is straightforward in the multiple-baseline algorithm, because this algorithm simply adds the SSD-in-inverse-depth instead of the disparity.

Next section, we demonstrate the effectiveness of using both horizontal and vertical baselines.

2. Horizontal and vertical baselines Experiment

Fig. 3 illustrates the experimental setup. The procedure is the same as the one in the horizontal baselines experiment, except images are taken by moving a camera horizontally and vertically. The acquisition parameters are shown in the last three rows ("Shrubbery2," "Corner," and "Guide") of table 1. The baseline length ranges from 20 mm for the closest camera pair to 60 mm for the farthest, which is somewhat shorter than the horizontal baselines case.

As shown in the last three rows in table 2, the number of the stereo pairs is 3 for each baseline and the total disparity range is 6 pixels.

2.1. Results

2.1.1. "Corner" data set

Fig. 14 shows the data set, together with an illustration of the arrangement of the camera and the objects.

Fig. 16 is the isometric plot of the depth map using three stereo pairs for each baseline orientation (six pairs in total). We observe that the building wall, especially the slanting part of the wall at the right is well estimated. The curb is separated from the shrubberies in the back and the road in the front. We can see the distances between the curb and the shrubberies.

For comparison, a depth map is computed using six stereo pairs in only horizontal orientation. Fig. 15 is the result. Many mismatches are observed at the wall and the curb, because the main features of these portions are horizontal.

2.1.2. SSD and SSSD in inverse depth

We examine the SSD and the SSSD of a point, such as a point on the wall or at the curb. Depth estimate of the point is correct using both horizontal and vertical baselines, though the estimate is incorrect using only horizontal baselines. Fig. 17 (a) and (b) show the SSD and the SSSD of the points ① and ② in fig. 16 respectively. Though the SSDs of the horizontal baselines do not show the clear minimum, the SSDs of the vertical baselines having a unique minimum at the correct point contribute to the determination of the correct minimum position of the total SSSD.

3. Comments

Parallelism

The computation of this algorithm is simple and local, which is suited for implementation on a massively parallel machine. We are implementing this algorithm on a MasPar, a Single Instruction Multiple Data (SIMD) machine with 4096 processors. At this moment, the processing time on MasPar is 0.9 second for producing a depth map with the 240x256 image size and the disparity range of 10 pixels, while it takes 51 seconds to do the same for SUN 4/40 (16 MIPS).

It is possible to implement this algorithm by a dedicated hardware or even on a chip for a real-time depth sensor.

Classification of depth measurements

We showed that the shapes of the SSD in-inverse-depth near the minimum of the SSSD may be useful to estimate the cause of mismatches like the occlusion and the sparse features cases. We expect that we can similarly analyze mismatches caused by a terminal edge or a highlight.

Acknowledgment

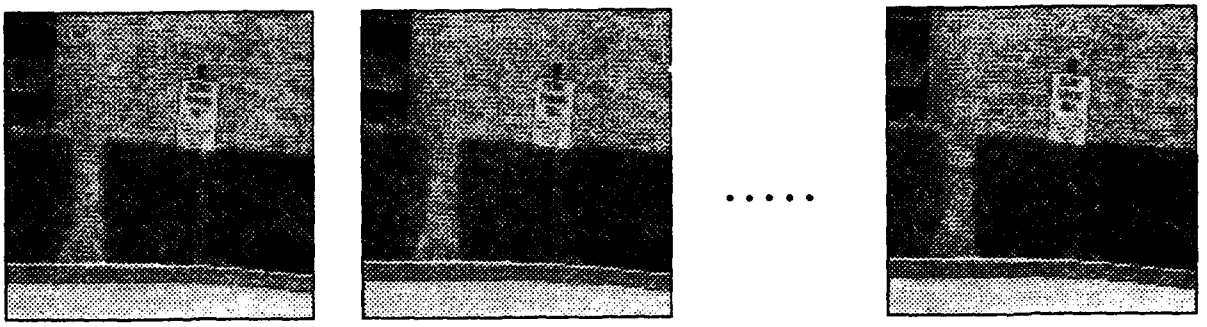
The authors would like to thank Kenichi Arakawa, Eric Krotkov, and Hans Thomas for assisting in image acquisition and reading the manuscript.

Table 2 Image processing

Name	Number of stereo pair	Image size	Disparity range
Town	3	256x240	4 - 14
Coal	5	256x240	30 - 40
Shrubbery	6	240x256	4 - 13
Parking meters	7	240x256	1 - 15
Sand	5	240x256	1 - 6
Shrubbery2	H : 3 V : 3	240x256	1 - 7
Corner	H : 3 V : 3	240x256	1 - 7
Guide	H : 3 V : 3	240x256	0 - 8

Table 1 Image acquisition

Name	Distance		Baseline length		TV camera	Focal length
	the nearest	the farthest	unit	longest		
Town	0.51m	1.02m	1.27mm	11.43mm	——	——
Coal	——	——	7.62mm	38.10mm	——	——
Shrubbery	19m	28m	19.05mm	114.10mm	SONY XC57	50mm
Parking meters	12m	34m	10.16mm	71.12mm	SONY XC57	50mm
Sand	6m	10m	2.54mm	12.70mm	SONY SSC-D7	50mm
Shrubbery2	19m	28m	20.0mm	60.00mm	SONY XC57	50mm
Corner	19m	28m	20.0mm	60.00mm	SONY XC57	50mm
Guide	16m	90m	20.0mm	60.00mm	SONY XC57	50mm



(a) 1 st (left most) (b) 2 nd (c) 7 th (right most)

Fig. 4 "Shrubbery" data

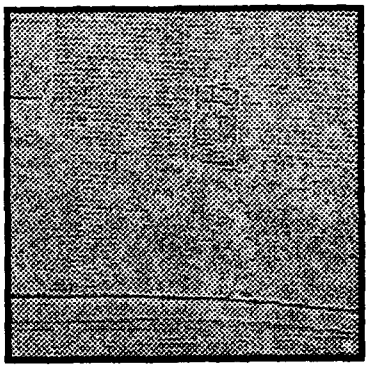


Fig. 5 Laplacian of Gaussian image

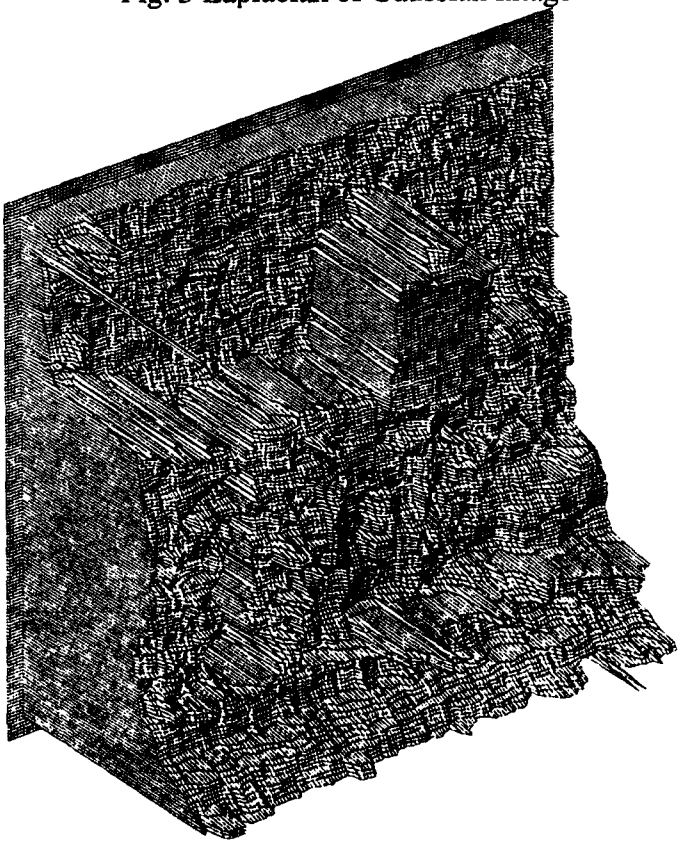


Fig. 6 Isometric plot of depth ("Shrubbery")

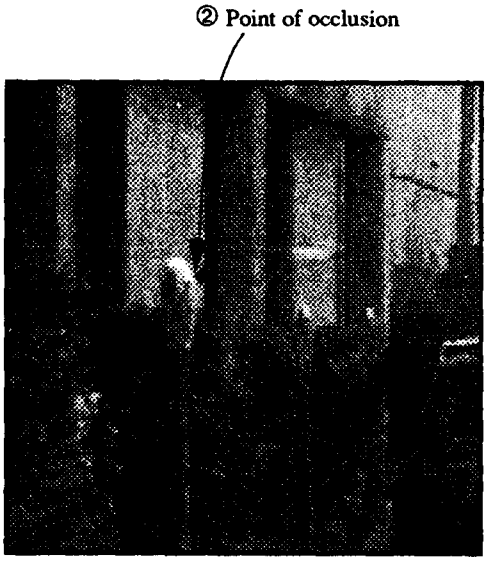


Fig. 7 "Parking meters" data

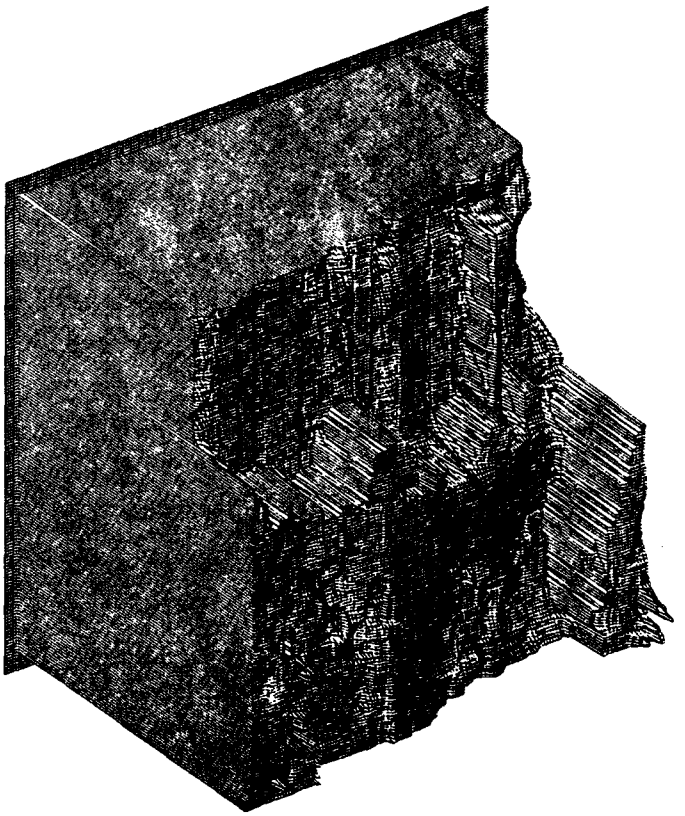


Fig.8 Isometric plot of depth ("Parking meters")

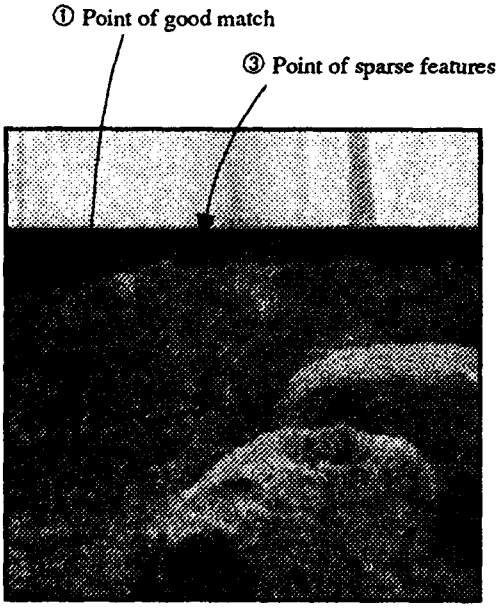


Fig. 9 "Sand" data

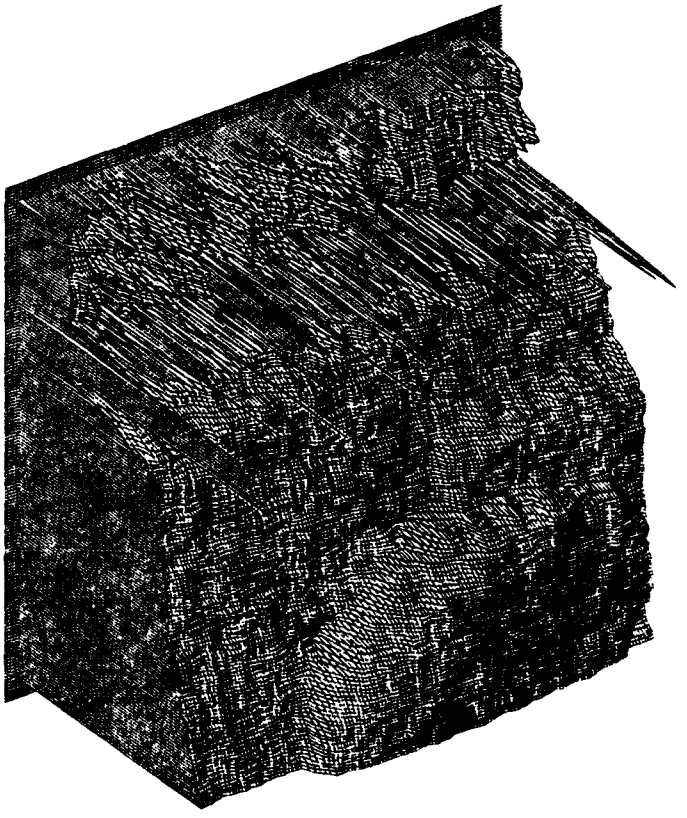
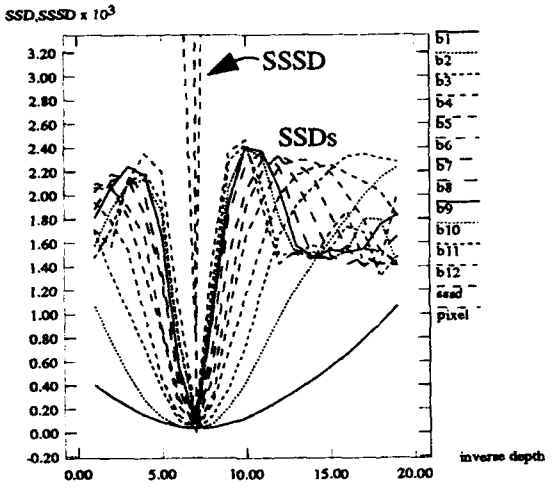
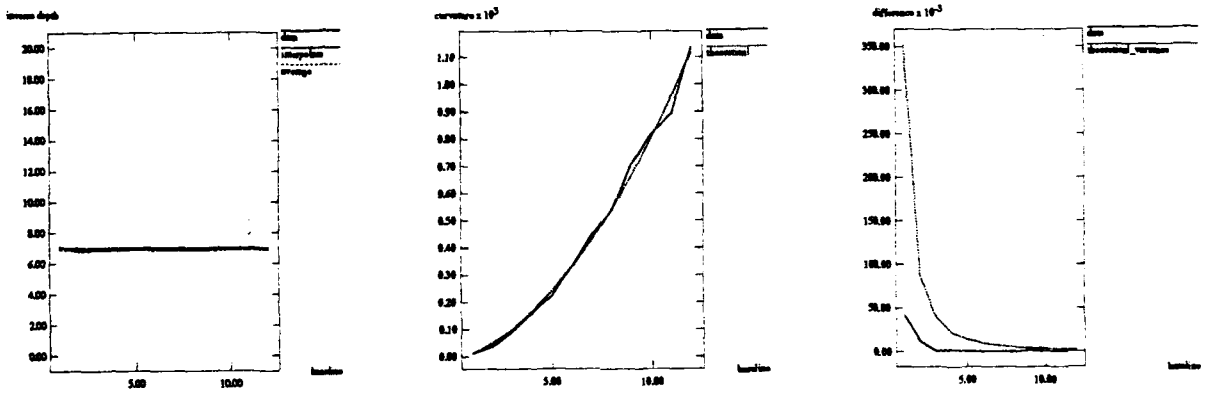


Fig.10 Isometric plot of depth ("Sand")



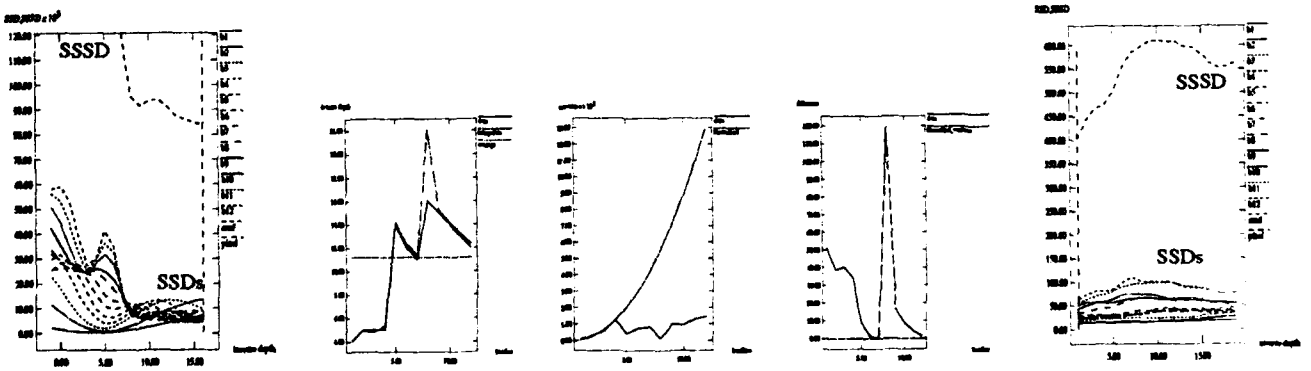
(a) Good match

Fig. 11 SSD and SSSD values vs. inverse depth



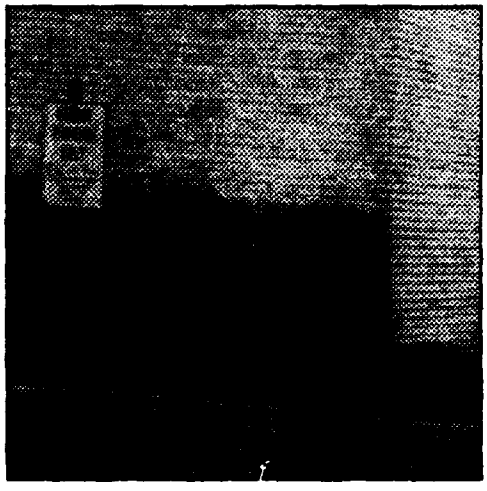
(a) Inverse depth (b) Curvature of SSD (c) Variance of differences

Fig. 12 Inverse depth, variance of differences between the minimum position of each SSD and the final estimate, and curvature of SSD from individual stereo pair near the minimum of SSSD vs. baseline (Case of a good match)

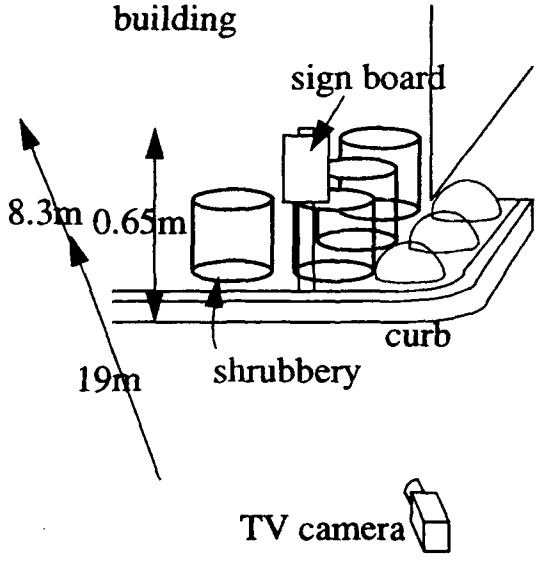


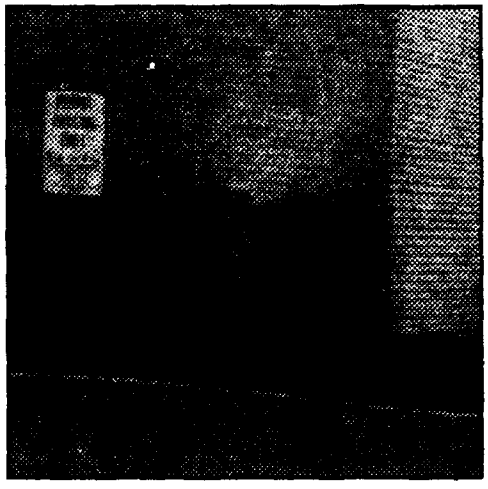
(a) Occlusion (b) Inverse depth (c) Curvature of SSD (d) Variance of differences (e) Sparse features

Fig. 13 SSD and SSSD values vs. inverse depth, inverse depth, variance of differences between the minimum position of each SSD and the final estimate, and curvature of SSD from individual stereo pair near the minimum of SSSD vs. baseline

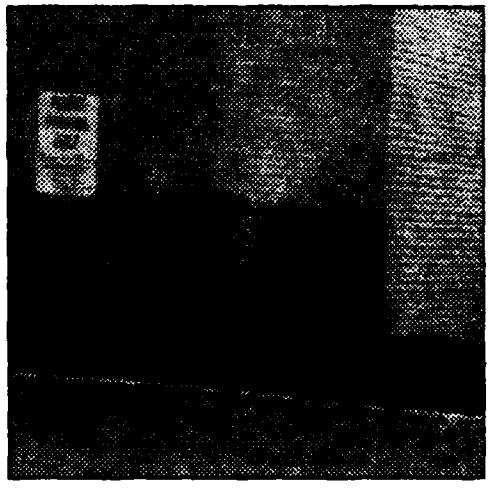


(a) Up most





(b) Left most



(c) Right most

Fig. 14 "Corner" data



Fig. 15 Isometric plot of depth resulted from horizontal baselines

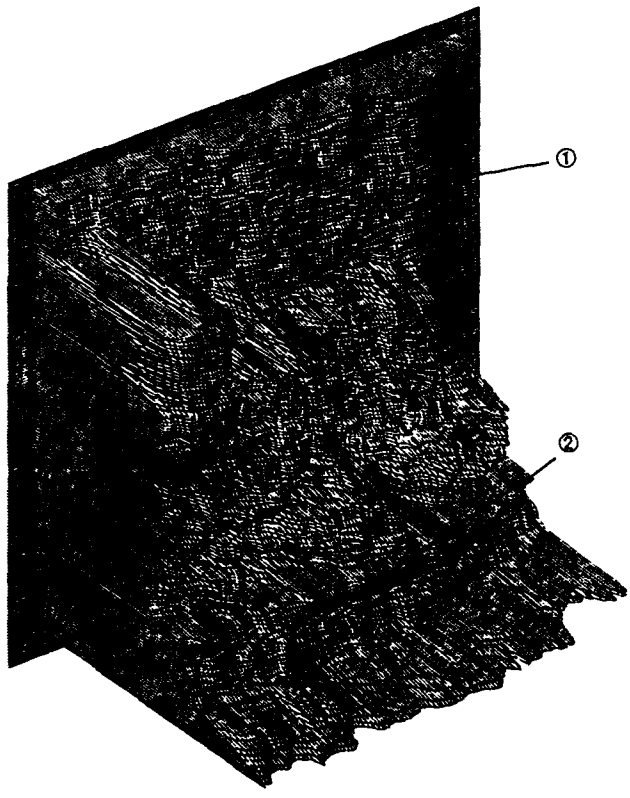
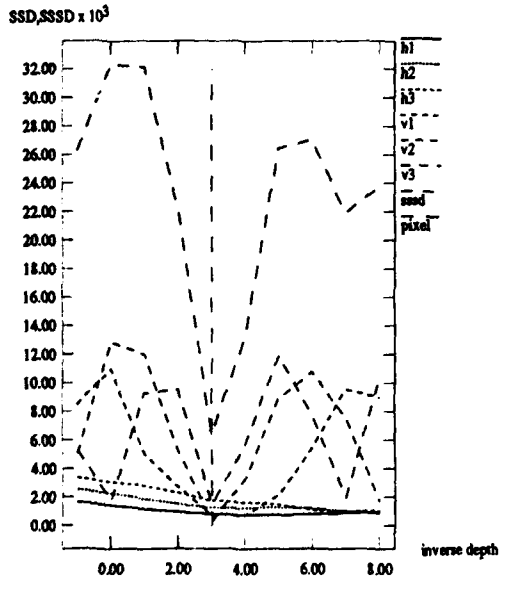
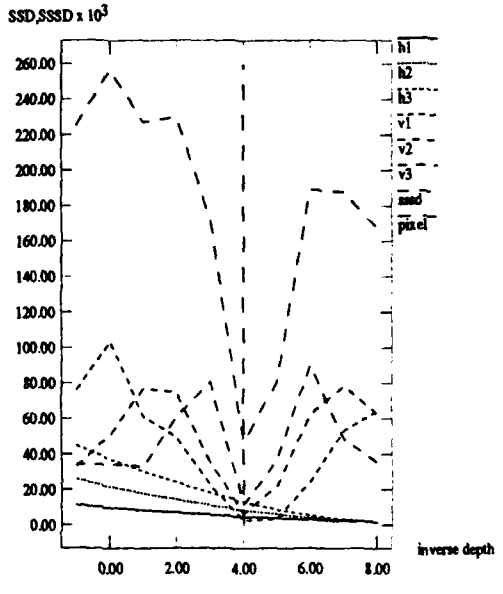


Fig. 16 Isometric plot of depth resulted from horizontal and vertical baselines



(a) On the wall



(b) At the curb

Fig. 17 SSD and SSSD values vs. inverse depth

Hierarchical Stereo Matching using Feature Groupings

V. Venkateswar
Texas Instruments
P.O. Box 655474, MS 238
Dallas, TX 75265

R. Chellappa
Dept. of Electrical Engineering
University of Maryland
College Park, MD 20742

Abstract

In this paper we design a feature based stereo matching system. We propose a hierarchical grouping process that groups line segments into more complex structures that are easier to match. The hierarchy consists of lines, vertices, edges and surfaces. Matching starts at the highest level of the hierarchy (surfaces) and proceeds to the lowest (lines). Higher level features are easier to match, because they are fewer in number and more distinct in form. These matches then constrain the matches at lower levels. Perceptual and structural relations are used to group matches into islands of certainty. A Truth Maintenance System (TMS) is used to enforce grouping constraints and eliminates inconsistent match groupings. The TMS is also used for reasoning in presence of uncertainty and to carry out belief revisions necessitated by additions, deletions and confirmations of hypotheses.

1 Introduction

Stereo matching is the process of fusing two images taken from different viewpoints to recover depth information in the scene. The process involves identifying corresponding features in two views and using their relative displacements together with camera geometry to estimate their depth. Line segments have gained popularity as features to be matched across scenes. The epipolar constraint [Barnard and Fischler, 1982] is used to restrict the number of line matches. Lines that can be matched must span the same set of epipolar lines. However, much ambiguity remains as a line in one scene can match several lines in the other scene, in spite of the epipolar constraint. So there is a necessity to consider a more global context to disambiguate line matches. Medioni and Nevatia [1985] use a minimum-differential-disparity criterion for disambiguation. Line matches that are adjacent and have similar disparity support each other in a relaxation framework. Ayache and Faverjon [1987] cluster adjacent matches with similar disparity. Both [Medioni and Nevatia, 1985] and [Ayache and Faverjon, 1987] are based on the idea that adjacent line segments should have similar disparity. This is, however, a weak constraint because adjacent lines in the scene need not have similar depth. This constraint favors frontoparallel surfaces [Hoff and Ahuja, 1989]. In this paper, we attempt to derive stronger constraints based on the topology of the scene to augment this weak constraint. Horaud and Skordas [1989] reduce matching ambiguity by also considering feature relations between line segments (collinear-with, same-junction-as, left-of, etc.) in the matching process. They construct an association graph [Ballard and Brown, 1982], whose nodes correspond to line matches and arcs are compatibility relations. Maximal cliques are extracted from this graph using the algorithm of Bolles and Cain [1982] and evaluated with a goodness measure to choose the best one. This procedure, however, is of exponential complexity. In this paper, we describe a polynomial time algorithm that forms local cliques instead of global ones. Local cliques are sufficient for disambiguation because of the hierarchical grouping process that we use.

The basic idea in our approach is to extract as much structure as possible from each scene before matching. This includes both topological and perceptual structure. We use a hierarchical grouping process to group line segments into complex structures. Relations between the structures are also computed. The hierarchy consists of *lines*, *vertices*, *surface edges* (edges, for short) and *surfaces*. The relations include *parallel*, *colinear*, *left-of* and *right-of*. The objective is to match the hierarchical relational graphs extracted in both scenes. Matching starts at the highest level (surfaces) and proceeds to the lowest (lines). At each level matches are grouped into strong clusters based on *structural* and *perceptual* relations. These groupings are then confirmed. Lim and Binford [Los Angeles CA 1987, Cambridge MA 1988] also match a hierarchy of features in their stereo system. Their hierarchy consists of edgels, curves, junctions, surfaces and bodies. But only those features that are grouped into bodies are matched and no ambiguity is assumed during matching. However, not all features can be grouped into higher level structures. For example, roads are characterized by parallel lines which cannot be joined at a junction. Also, our experience with several images shows that ambiguity is a serious problem, in spite of using a feature hierarchy. To deal with ambiguity as to the existence of both the features and their matches, we treat them as hypotheses whose beliefs can be later revised (to true or false) as more evidence accumulates. When these hypotheses are grouped, they form contexts. Constraints are specified by the user as *nogoods*. These constraints identify certain contexts as contradictory. A Truth Maintenance System (TMS)

manages this search space made up of hypotheses, their groupings (contexts) and constraints. The TMS ensures that the final solution contains no contradictions. Chung and Nevatia [1991] also match a similar hierarchy. A constraint satisfaction network [Mohan and Nevatia, 1989] is used to deal with ambiguities. However this relaxation approach is only an approximation technique and there is a danger of getting stuck in local minima.

Both numerical and symbolic approaches are known in the literature to deal with uncertainty. For a review, see [Bhatnagar and Kanal, 1988]. Symbolic approaches include TMSs [Doyle, 1979, de Kleer, 1986, Martins, 1991]. In TMSs, uncertainty is dealt with by making *assumptions* (*hypotheses*) whose beliefs can be later revised as new evidence accumulates. Unlike numerical methods, symbolic methods cannot represent degrees of conflict (but see [Laskey and Lehner, 1989 90], which integrates Dempster-Shafer formalism into a TMS), but they permit explicit reasoning about the assumptions that actually led to the conflict (conflict resolution). Since, various constraints, like the uniqueness constraint, ordering constraint etc., are strictly (and not to a degree) enforced in most stereo algorithms (including the one presented in this paper) there is more of a need for an approach that can reason about the conflicts produced by these constraints than one that can deal with degrees of conflict. So we choose to use a TMS to deal with uncertainty.

Constraint satisfaction techniques like backtracking [Grimson, 1990b, Grimson, 1990a], discrete relaxation [Mackworth, 1977, Mackworth and Freuder, 1985, Medioni and Nevatia, 1984] or continuous relaxation [Mohan and Nevatia, 1989, Price, 1985] can also be used to sift through the space of competing and supporting matches. The advantage in using a TMS for the image matching problem is that binary, ternary or higher order constraints are easily incorporated. Also, the problem solver has more control of the path to solution and there is a possibility of avoiding local minima. Further, no backtracking is involved in the assumption-based TMS (ATMS) [Inf, 1987] we use in our system. Instead we do a simultaneous search in a context lattice. The maladies associated with backtracking have been well documented [de Kleer, 1986, Mackworth, 1977]. In particular, inconsistencies are repeatedly rediscovered, as there is no concept of remembering mistakes.

2 A Feature Hierarchy

The proposed hierarchy, shown in Figure 1, consists of lines (straight line segments), vertices (junctions of line segments), edges (collections of colinear lines with vertex terminations) and surfaces (contiguous sets of edges, i.e. edge-rings). Surfaces can be open or closed. Currently objects are not included in our hierarchy, mainly because in most of the images that we encountered, surfaces were the most complex structures visible. But the extension is straight forward. The domain of applicability is any scene whose contours can be approximated with a series of straight line segments (they need not lie on a plane). This includes urban, indoor and factory scenes.

First, edgels (edge pixels) in the image are extracted using the Canny edge operator [Canny, 1986]. A modified connected-components algorithm [Venkateswar and Chellappa, 1990] is used to link the edgels into straight line segments. Perceptual relations between the straight line segments (lines, for short) are then determined. For any line, only those lines within a local neighborhood (typically 5% the width of the picture) are tested for perceptual relations. These include *parallel, colinear* and *proximate* relationships. For each line, only the closest parallel, colinear and proximate line relationships (one on each side) are retained. Bucketing techniques [Ayache and Faverjon, 1987, Knuth, 1973] are used to reduce the number of comparisons necessary to compute these relations between lines. Two lines whose end points are close to each other are possibly connected in the scene. A *vertex* is hypothesized to join the two lines in an L-junction. The position of the vertex is at the intersection that is obtained if the lines were extended. *Edges* are sets of colinear line segments with vertex terminations. They represent surface boundaries in the scene that are fragmented into lines (because of poor contrast, etc). For extracting edges; if two vertices in the scene have lines that are colinear and face each other, an edge is hypothesized to join the vertices. This procedure generates several spurious edges, because of accidental alignment of vertices. To eliminate such edges, all lines that overlap each edge are accumulated. Then the percentage overlap of each edge with these lines is calculated. This gives the line segment support for each edge. All the edges with insufficient line segment support (less than 90%) are deleted. A few spurious edges may still remain. However, it is unlikely that such edges will have a corresponding match in the other scene. A surface (edge-ring) is an ordered set of connected edges (at least two) that cannot be extended any further.

The initial line data given to the system is noisy. The end points are not well localized and some lines are fragmented into segments. This is a direct result of the noisy nature of real images and the fact that low level techniques, including edge pixel detection and line extraction, are imperfect. This leads to uncertainties when grouping lines into higher level features. To deal with these uncertainties, features in the hierarchy (other than lines) are conjectured as *hypotheses*. These features may be *confirmed* when more evidence becomes available, including evidence of a matching feature in the other image. The hypotheses and their groupings define a context graph (a directed acyclic graph). This is illustrated in Figure 2 which shows the context graph for a simple stereo pair. The context graph is the search space of the problem. Nodes in the graph correspond to collections of hypotheses. The objective is to incrementally build this graph and identify those nodes that correspond to partial solutions.

3 Matches and Match Groupings

The feature hierarchy together with feature relationships forms a relational graph (or a semantic network). The objective is to match the graphs derived from a stereo pair. We proceed by finding matches for each feature and

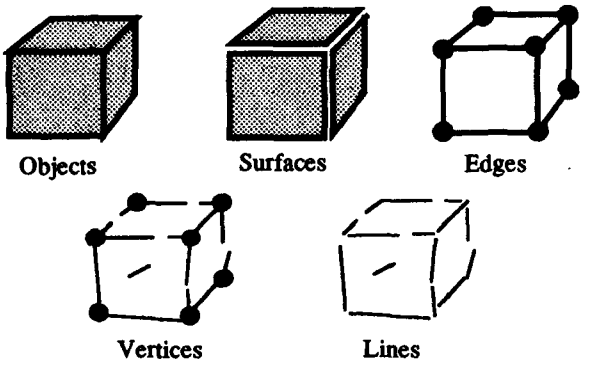
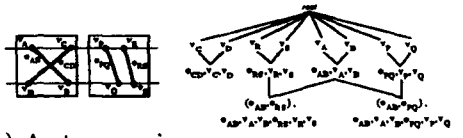


Figure 1: A feature hierarchy



(a) A stereo pair.

(b) Context graph

Figure 2: Hypotheses and their groupings form a context graph. v_A, v_B, v_C, v_D, v_P, v_Q, v_R and v_S are vertex hypotheses. e_{AB}, e_{CD}, e_{PQ} and e_{RS} are edge hypotheses. (e_{AB}, e_{PQ}) and (e_{AB}, e_{RS}) are edge-match hypotheses.

grouping these matches into local cliques. The matching is hierarchical, we start by matching at the surface level and proceed to the line level. For two features to match, their component features must match, recursively. For example; two surfaces match only if their edges match (in the same order), their edges match only if their vertices match and so on. At each level those features that were not matched at previous levels (as component features) participate in the matching process. Below we detail the unary constraints used before hypothesizing a match between features.

epipolar constraint: For each feature in one image, we search for a matching feature in the other image, within the epipolar lines spanning the feature. To account for uncertainty in the positions of the features, we tolerate a deviation of about 4 pixels perpendicular to the epipolar lines. Lines are sometimes fragmented, so a small overlap (25%) of epipolar extents is considered sufficient for matching.

area and length constraints: We expect the areas of matching surfaces to be within 75% of each other. The lengths of matching edges must be within 75% of each other and those of lines must be within 25% (to account for their fragmentation).

orientation constraint: Since the two views in a stereo pair are not very far apart we expect the orientations of features to be similar [Arnold and Binford, 1980]. So we bound the orientation difference between matching features at 30°. Similar bounds have been used in other work [Medioni and Nevatia, 1985, Horaud and Skordas, 1989]. For two vertices that satisfy the epipolar constraint, the orientation constraint applies to their matching lines. Further, the matched lines must be in the same relative order (clockwise or anti-clockwise).

contrast constraint: For two linear features (lines or edges) to match, their contrast must be similar. Each line and edge marks a boundary between a dark and a bright region. If the bright region is on the same side of the linear features (to the left or right) then they can be matched.

For each feature in one image there may be several match hypotheses in spite of the unary constraints detailed above. A more global context is needed to disambiguate between competing matches. Ambiguity is least at the surface level where features have the most structure and is maximum at the line level, where typically a line has several possible matches. So we start by first matching at the surface level. To reduce ambiguity even further, we group compatible matches into local cliques. Compatibility is based on consistent structural and perceptual relations across the scenes (Figure 3). Structural groupings are based on unmatched features at the previous level of the hierarchy. These unmatched features are used as foci-of-attention when matching. For example, consider the groupings when matching at line level. Figure 3 shows a typical case. Vertex v_{AB} has no match in the other image. But its lines l_A and l_B match l_P and l_Q respectively. Further, these lines when extended intersect on the epipolar line of v_{AB}. This is consistent with the presence of a vertex at this location. This vertex was not detected earlier during the feature grouping process because the lines l_P and l_Q were not closer than a threshold. Unmatched vertices serve as foci-of-attention when matching at the line level. The system focuses on these vertices and creates structural groupings. The matches (l_A, l_P) and (l_B, l_Q) together with vertex v_{AB} form a structural grouping. Perceptual groupings are based on consistent perceptual relations [Lowe, 1987] across the scenes (in case of surfaces and vertices, perceptual relations are based on relations of their constituent edges and lines respectively). These include *parallel*, *colinear* and *proximate* relations. In Figure 3, l_C and l_D as well as their matches l_R and l_S are connected through the *parallel* relation. So the match pairs (l_C, l_R) and (l_D, l_S) form a parallel perceptual grouping.

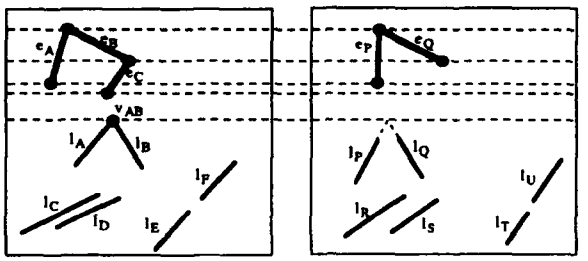


Figure 3: Match Groupings

Those groupings that share a match are formed into super-groups. Groupings are currently restricted to a size of three consistent matches. These groupings correspond to partial solutions to the matching prob-

lem. So our search graph is only three levels deep. This reduces matching complexity. We have found that these groupings (together with the hierarchical feature grouping process) are usually sufficient to completely disambiguate between competing matches, i.e. it is hard to find consistent groupings for an erroneous match hypothesis. At each level structural groupings are confirmed first, followed by perceptual groupings. Among perceptual groupings, proximate groupings are confirmed last. Proximate groupings are those matches that have features that are *proximate* in both scenes and have a similar disparity. However, the constraint that adjacent features in the scene should have similar disparity is a weak constraint, so these groupings get the least priority. Other researchers have chosen to use the similar disparity criterion as their primary constraint [Medioni and Nevatia, 1985, Ayache and Faverjon, 1987]. Larger groupings are confirmed before smaller groupings (groups of three first, followed by groups of two and then ungrouped matches) because they have a more global context. Within each size the confirmation is completely random.

When a grouping of match hypotheses is *confirmed* the beliefs of several hypotheses are affected and must be revised. The process of *belief revision* is handled by the TMS. First, all underlying hypotheses (both feature and match hypotheses) of the confirmed grouping must be converted to truths. This is done by collapsing the portion of the context graph that supports the grouping into the root context. These root hypotheses then are made visible in all contexts where they are grouped with the other hypotheses. This may lead to some contradictory groupings. The contexts corresponding to such groupings are eliminated by the TMS. The strategy of confirming partial solutions and eliminating contradictory contexts has the effect of preventing unbridled growth in the size of the context graph. For example; consider the context graph shown in Figure 2. Suppose the match hypothesis (single grouping) (e_{AB}, e_{PQ}) is confirmed. Then, the following actions take place.

1. The portion of the context graph supporting this grouping is collapsed into the root. The result is shown in Figure 4(a). The root hypotheses v_A, v_B, v_P, v_Q, e_{AB}, e_{PQ} and (e_{AB}, e_{PQ}) are now visible in all contexts by inheritance (but for clarity are not shown explicitly in the non-root contexts of the figure).
2. This creates two contradictory contexts. The context with match hypothesis (e_{AB}, e_{RS}) is invalid because it violates the match (e_{AB}, e_{PQ}) (uniqueness constraint, see Section 4). Similarly, the context with edge hypothesis e_{CD} is invalid because this edge intersects true edge e_{AB}. Both contexts are eliminated, resulting in the context graph, shown in Figure 4(b).

The effect is that when a grouping is confirmed those hypotheses that violate its elements (either feature or match) are eliminated. In the final set of matches in the root context there will be no contradictions. These matches correspond to a solution to the stereo matching problem.

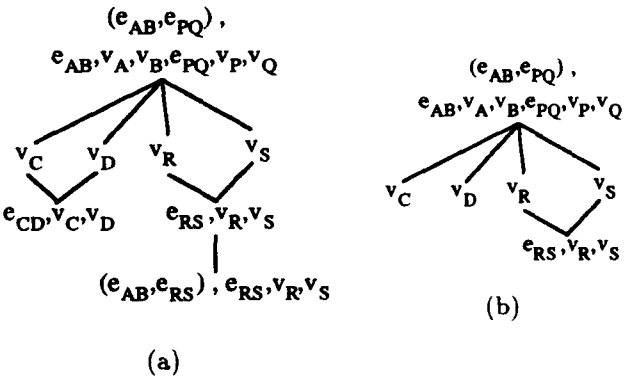


Figure 4: Confirming a context.

As described earlier, matching is defined recursively, so that when two features at a level are matched, so are their constituent features at the lower levels. Matching at one level of the hierarchy affects matching at lower levels. For example, consider the effect of surface matches on line matches. When two surfaces are matched, the line segments constituting their boundaries are also matched. When the surface match is confirmed so are the corresponding line matches. This affects the matching at the line level in three ways. First, there are fewer lines left to be matched at the line level. Second, any new line match that violates confirmed line matches is eliminated by the TMS. Third, the confirmed line matches serve as seeds for match groupings at the line level.

4 Grouping Constraints

Two classes of groupings result during the grouping and confirming procedure described in the previous section. *Direct* groupings are those that result when hypotheses are grouped directly based on structural and perceptual relations. *Indirect* groupings are those that result when contexts are confirmed. The underlying hypotheses of confirmed contexts are asserted into the root and are inherited down the context hierarchy into all the contexts. These hypotheses are then grouped with the other hypotheses in these contexts.

Certain groupings of features or matches are contradictory. Our objective is to eliminate all contradictory groupings (whether direct or indirect). As an example of a contradictory grouping, consider the intersecting edge hypotheses e_{AB} and e_{CD} shown in Figure 2. Physical edges in the scene cannot intersect, so these edge hypotheses are mutually contradictory. Of course, this is not to say that two edges in a scene cannot cross each other when viewed from some viewpoint. But if they did, part of one edge will be occluded by the surface formed by the other and this edge will not be detected by the feature grouping process. Any context with a grouping of intersecting edge hypotheses is therefore a *nogood*. Such *nogood* contexts are specified to the system explicitly using *nogood pattern combinations* (nogoods, for short). These are sets of patterns that identify contradictory groupings of hypotheses. This is an elegant way of spec-

ifying constraints. Any context with propositions that match all the patterns of a nogood is eliminated by the TMS. The effect is that in the final solution there are no contradictions. Below we illustrate some of the grouping constraints used in the system.

uniqueness constraint: Any feature can match at most one feature. This follows from the fact that the image of a point in the scene will project onto a single point in each view. The uniqueness constraint for edge matches is illustrated in Figure 5. This is specified as the following nogood.

```
(nogood
  (edge-match ?eX ?eY)   ;; if ?eX matches ?eY
  (edge-match ?eX ?eZ))  ;; and also matches ?eZ
```

This constraint needs to be modified for lines, which tend to be fragmented. In this case, if a line matches more than one line, we expect those matches to be colinear.

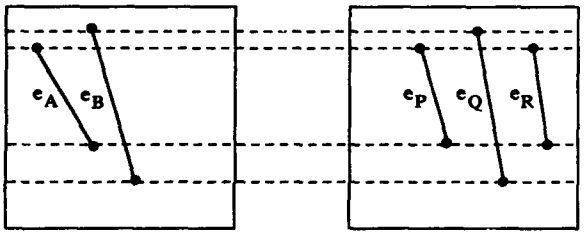


Figure 5: Uniqueness and ordering constraints for edges: The edge match hypotheses (e_A, e_P) and (e_A, e_R) cannot be grouped because e_A can match only one edge (uniqueness constraint). (e_A, e_R) and (e_B, e_Q) are inconsistent because e_A is to the left-of e_B, but e_R is to the right-of e_Q. This violates the ordering constraint.

ordering constraint: This constraint was originally used by Baker and Binford [Baker and Binford, Palo Alto CA 1982] and precludes any order reversal when matching along an epipolar line (Figure 5). It is satisfied in most natural scenes. Wires and overhanging surfaces may, however, create problems [Baker and Binford, Palo Alto CA 1982]. Most stereo systems enforce this constraint strictly [Baker and Binford, Palo Alto CA 1982, Hsieh *et al.*, 1990, Horaud and Skordas, 1989]. The advantage is that the constraint reduces the search space drastically at the expense of missing a few correct matches.

topological constraints: The topology of the feature hierarchy is viewpoint insensitive and we do not expect it to change between scenes. Topological constraints rule out several contexts. As one example, consider the matches (e_A, e_P) and (e_B, e_Q) in Figure 6. The matches are connected in one scene and disconnected in the other. So their grouping is inconsistent. Two more examples are shown in the same figure. The last two examples show ternary constraints involving three hypotheses.

The constraints described in this section are binary (in some cases ternary) constraints [Grimson, 1990a] that

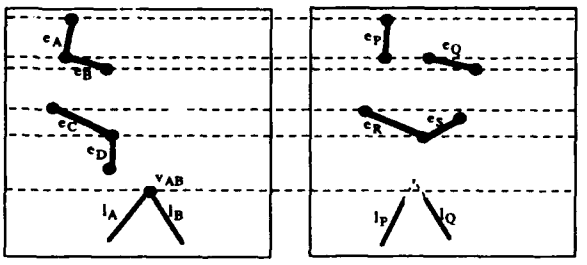


Figure 6: **Topological constraints:** The edge match hypotheses (e_A, e_P) and (e_B, e_Q) are incompatible because they are connected in one image and not in the other. Similarly, the edge match hypothesis (e_C, e_R) is incompatible with the edge hypotheses e_D and e_S because these edge hypotheses do not match and we expect topology to be preserved across scenes. The vertex hypothesis v_{AB} and the match hypotheses (l_A, l_P) and (l_B, l_Q) are incompatible because l_P and l_Q do not intersect on the epipolar line of v_{AB} when extended.

prohibit certain groupings of matches. The epipolar, orientation, length, area and contrast constraints described in Section 3 are unary constraints that help in reducing the number of matches that are hypothesized and later subjected to the grouping process.

4.1 Matching Complexity

In this section we do an approximate analysis of the matching complexity. We start by analyzing the complexity at one level of the hierarchy and later analyze the multi-level case. Suppose there are N features at a particular level of the hierarchy and E_l epipolar lines in each image. The number of features per epipolar line is $\frac{N}{E_l}$. Features are matched across epipolar lines. So the number of feature pairs to be tested for matching across each epipolar line is $\frac{N^2}{E_l^2}$. Since there are E_l of these epipolar lines, the number of feature pairs to be tested for the entire image is

$$\frac{N^2}{E_l}. \tag{1}$$

Suppose a fraction f of these satisfy the orientation and contrast constraints. Then the final number of matches will be $M = f\frac{N^2}{E_l}$.

We now group these match hypotheses based on structural and perceptual relations. For simplicity of analysis, assume that there are $\frac{M}{3}$ groupings of size 3. Binary and ternary constraints are evaluated for each grouping. The complexity of these constraint evaluations is proportional to

$$\frac{M}{3} = \frac{fN^2}{3E_l}. \tag{2}$$

These groupings are confirmed at random. When a grouping is confirmed all matches comprising the grouping are converted to truths and asserted into the root context. These matches are then inherited down the context tree and they must be compared with the other matches outside the root context to check for violations

of grouping constraints. The first confirmation leaves $(M-3)$ match hypotheses outside the root context, the second $(M-6)$ and so on, till all groupings are confirmed (assume for simplicity that no grouping is eliminated because of constraint evaluations). The complexity of the confirmation process is proportional to

$$(M-3)+(M-6)+(M-9)+\ldots = \frac{M^2}{6} - \frac{M}{2} \approx \frac{M^2}{6} \quad (3)$$

From (1), (2) and (3) it can be seen that the time for matching is essentially decided by (3), the confirmation process. This was also empirically observed.

The above analysis shows that the complexity of matching at each level is $O(N^4)$, where N is the number of features at that level. So if only lines were matched (i.e. there was no hierarchy) the complexity is $O(l^4)$, where l is the number of lines in each frame. We now examine how the hierarchical matching procedure reduces the proportionality constant of this complexity (complexity will still be proportional to l^4). Let the reduction in number of feature hypotheses between consecutive levels be k_1. Let the fraction of reduction in the number of features (not hypotheses) between consecutive levels be k_2. k_1 is smaller than k_2 ($k_1 \leq k_2$). The inequality is because not all feature hypotheses may participate in the feature grouping process. The table in Figure 7 assists in computing the complexity reduction. The first column shows the names of features. The second column shows the number of features at each level, assuming there are l lines at the line level and a fractional reduction of k_1. The third column shows the number of features that are actually considered for matching. For example, consider the s_m surfaces matched at the surface level. These correspond to $\frac{f}{k_2} s_m$ edge matches, $\frac{f}{k_2^2} s_m$ vertex matches and $\frac{f}{k_2^3} s_m$ line matches. These matched features will not be considered for matching again. This leads to a reduction in the number of features participating in matching at lower levels. The third column shows the original number of features decremented by the features already matched at previous levels. So the hierarchical matching strategy reduces complexity by a factor of $\frac{l_m^4+v_m^4+e_m^4+s_m^4}{l^4}$, where the numerator sums the computations performed at different levels of the hierarchy. As a typical example, if $f = 0.6$ (60% of features at each level matched), $k_1 = 0.4$ (each level has 0.4 times the number of hypotheses at its lower level) and $k_2 = 0.5$ (fractional reduction in number of features across two levels in the hierarchy) the complexity reduction factor is 0.01. So the hierarchical matching procedure significantly reduces matching complexity. Further, it also ensures that the resultant matches are more correct. Both factors are important.

complexity analysis		
feature	*no.*	*no. to be matched*
lines	l	$l_m = l - \frac{f}{k_2^3} s_m - \frac{f}{k_2^2} e_m - \frac{f}{k_2} v_m$
vertices	$k_1 l$	$v_m = k_1 l - \frac{f}{k_2^2} s_m - \frac{f}{k_2} e_m$
edges	$k_1^2 l$	$e_m = k_1^2 l - \frac{f}{k_2} s_m$
surfaces	$k_1^3 l$	$s_m = k_1^3 l$

Figure 7: Table for computing matching complexity

5 Examples

The stereo algorithm is illustrated first for the building scene shown in Figure 8. The lines and vertices (dark circles) extracted from these images are shown in Figure 9. The vertices are grouped into edges (shown in Figure 10 as dark lines). No surfaces derived from the images match exactly. However the unmatched surfaces are useful as they are used as foci of attention when matching. Results after matching edges are shown in Figure 11. The vertices and lines of matched edges are also matched. These matches constrain the matches at lower levels. For example, any line match hypothesis that violates the ordering constraint with respect to these confirmed line matches will be negated. Figure 12 shows the feature matches after matching at the vertex level. To demonstrate the use of focus of attention, we show the results after matching lines by focusing on unmatched vertices (Figure 13). The final result after matching lines is shown in Figure 14. Figures 15 and 16 show labelled vertex and line matches. Matching vertices and lines in these figures are given the same label.

Next, we illustrate the matching process for a stereo pair of images of a portion of LAX airport (Figure 17). Figures 18 and 19 show the feature hierarchy. Results of matching surfaces are shown in Figure 20. Final edge matches are shown in Figure 21. Vertex matches are shown in Figure 22. Figure 23 shows the final matches at the line level. Figures 24 and 25 show labelled vertex and line matches.

6 Acknowledgment

The work reported here was partially supported by the Joint Services Electronics Program, at the University of Southern California through the Air Force office of Scientific Research under Contract F-49620-88-C-0067. The second author is also affiliated with the Center for Automation Research and the Institute for Advanced Computer Studies at the University of Maryland.

References

[Arnold and Binford, 1980] R. D. Arnold and T. O. Binford. Geometric constraints in stereo vision. In *Proceedings, SPIE Vol.238-Image Processing for Missile Guidance*, pages 281–292, 1980.

[Ayache and Faverjon, 1987] N. Ayache and B. Faverjon. Efficient registration of stereo images by matching graph descriptions of edge segments. *International Journal of Computer Vision*, 1:107–131, 1987.

[Baker and Binford, Palo Alto CA 1982] H. H. Baker and T. O. Binford. A system for automated stereo

mapping. In *Proceedings, DARPA Image Understanding Workshop*, pages 215–222, Palo Alto, CA, 1982.

[Ballard and Brown, 1982] D. H. Ballard and C. M. Brown. *Computer Vision.* Prentice–Hall, Englewood Cliffs, New Jersey, 1982.

[Barnard and Fischler, 1982] S. T. Barnard and M. A. Fischler. Computational stereo. *Computing Surveys*, 14:553–572, 1982.

[Bhatnagar and Kanal, 1988] R. K. Bhatnagar and L. N. Kanal. Handling uncertain information: A review of numeric and non-numeric methods. In L. N. Kanal and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 3–26. Elsevier Science Publishers, Amsterdam, The Netherlands, 1988.

[Bolles and Cain, 1982] R. C. Bolles and R. A. Cain. Recognizing and locating partially visible objects: The local-feature-focus method. *International Journal of Robotics Research*, 1(3):57–82, 1982.

[Canny, 1986] J. F. Canny. A computational approach to edge detection. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI–8:679–698, 1986.

[Chung and Nevatia, 1991] R. C. K. Chung and R. Nevatia. Use of monocular groupings and occlusion analysis in a hierarchical stereo system. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition CVPR '91*, pages 50–55, Maui, Hawaii, June 1991.

[de Kleer, 1986] J. de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[Doyle, 1979] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[Grimson, 1990a] W. E. L. Grimson. The combinatorics of object recognition in cluttered environments using constrained search. *Artificial Intelligence*, 44:121–166, 1990.

[Grimson, 1990b] W. E. L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints.* The MIT Press, Cambridge, Massachusetts, 1990.

[Hoff and Ahuja, 1989] W. Hoff and N. Ahuja. Surfaces from stereo: Integrating feature matching, disparity estimation, and contour detection. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-11:121–136, 1989.

[Horaud and Skordas, 1989] R. Horaud and T. Skordas. Stereo correspondence through feature grouping and maximal cliques. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-11:1168–1180, 1989.

[Hsieh *et al.*, 1990] Y. C. Hsieh, F. Perlant, and D. M. McKeown, Jr. Recovering 3D information from complex aerial imagery. In *Proceedings International Conference on Pattern Recognition, Vol. 1*, pages 136–146, Atlantic City, NJ, June 1990.

[Inf, 1987] Inference Corporation, Los Angeles. *ART reference manual*, January 1987.

[Knuth, 1973] D. E. Knuth. *The Art of Computer Programming, Vol. 3.* Addison Wesley, Reading, Massachusetts, 1973.

[Laskey and Lehner, 1989 90] K. B. Laskey and P. E. Lehner. Assumptions, beliefs and probabilities. *Artificial Intelligence*, 41:65–77, 1989-90.

[Lim and Binford, Cambridge MA 1988] H. S. Lim and T. O. Binford. Structural correspondence in stereo vision. In *Proceedings, DARPA Image Understanding Workshop*, pages 794–808, Cambridge, MA, 1988.

[Lim and Binford, Los Angeles CA 1987] H. S. Lim and T. O. Binford. Stereo correspondence: A hierarchical approach. In *Proceedings, DARPA Image Understanding Workshop*, pages 234–241, Los Angeles, CA, 1987.

[Lowe, 1987] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[Mackworth and Freuder, 1985] A. K. Mackworth and E. C. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25:65–74, 1985.

[Mackworth, 1977] A. K. Mackworth. Consistency in networks on relations. *Artificial Intelligence*, 8:99–118, 1977.

[Martins, 1991] J. P. Martins. The truth, the whole truth, and nothing but the truth: An indexed bibliography to the literature of truth maintenance systems. *AI Magazine*, 11(5):7–25, 1991.

[Medioni and Nevatia, 1984] G. Medioni and R. Nevatia. Matching images using linear features. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-6:675–685, 1984.

[Medioni and Nevatia, 1985] G. Medioni and R. Nevatia. Segment-based stereo matching. *Computer Vision, Graphics and Image Processing*, 31:2–18, 1985.

[Mohan and Nevatia, 1989] R. Mohan and R. Nevatia. Using perceptual organization to extract 3-d structures. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI–11:1121–1139, 1989.

[Price, 1985] K. E. Price. Relaxation matching techniques–a comparison. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI–7:617–623, 1985.

[Venkateswar and Chellappa, 1990] V. Venkateswar and R. Chellappa. Extraction of straight lines in aerial images. In *Proceedings, Fifth European Signal Processing Conference*, pages 1671–1674, 1990.

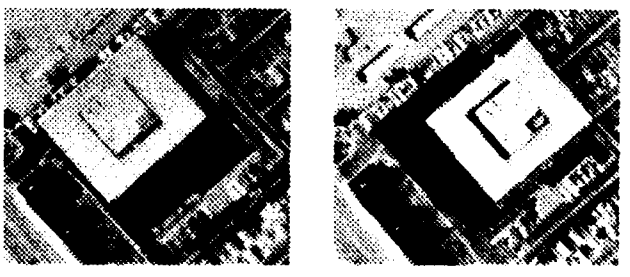


Figure 8: Right view and left view of a building image.



Figure 9: Lines and vertices.

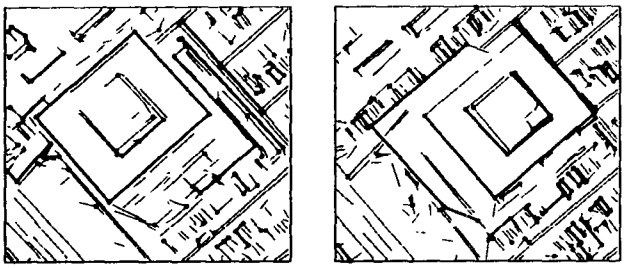


Figure 10: Edges (dark lines).

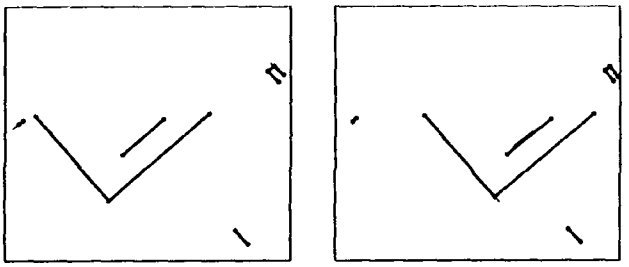


Figure 11: After matching at edge level.

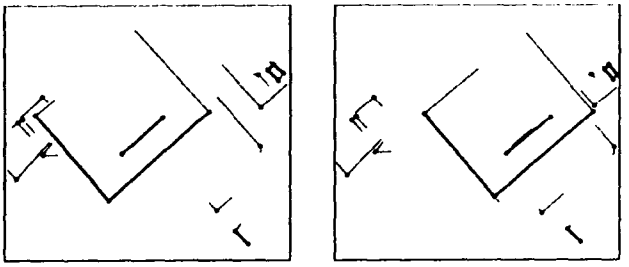


Figure 12: After matching at vertex level.

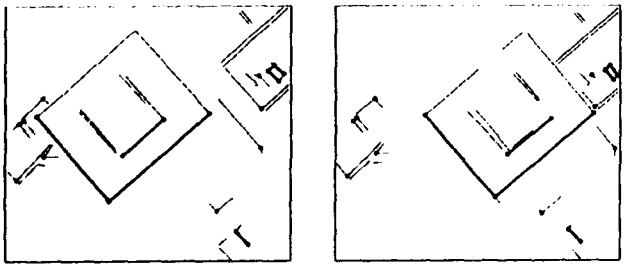


Figure 13: After matching lines by using unmatched vertex features as foci of attention.

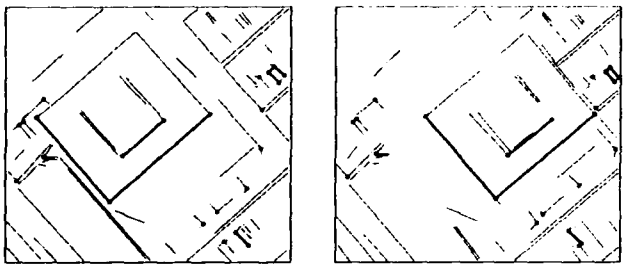
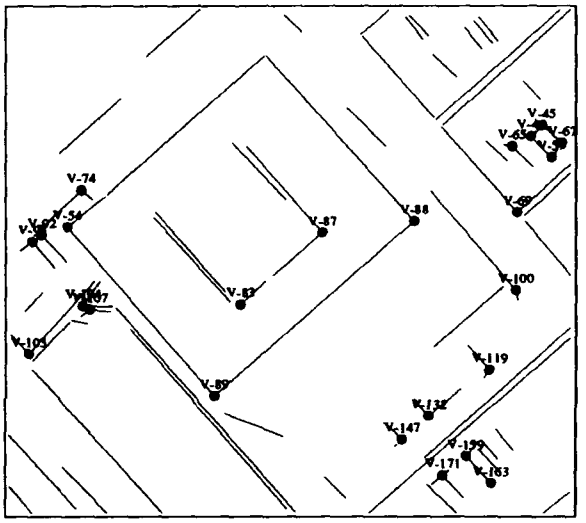
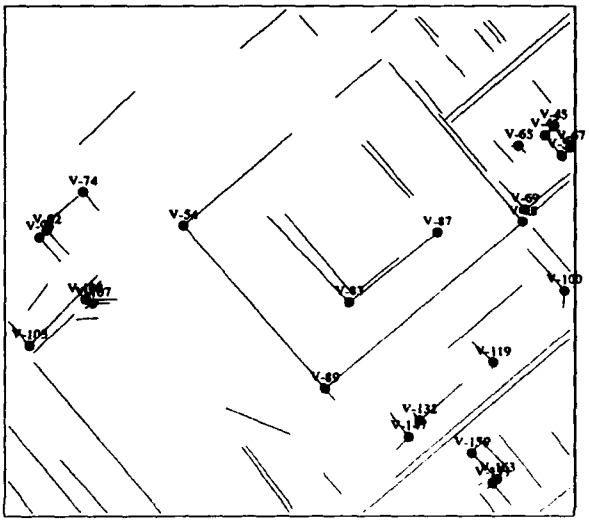


Figure 14: Final matches.

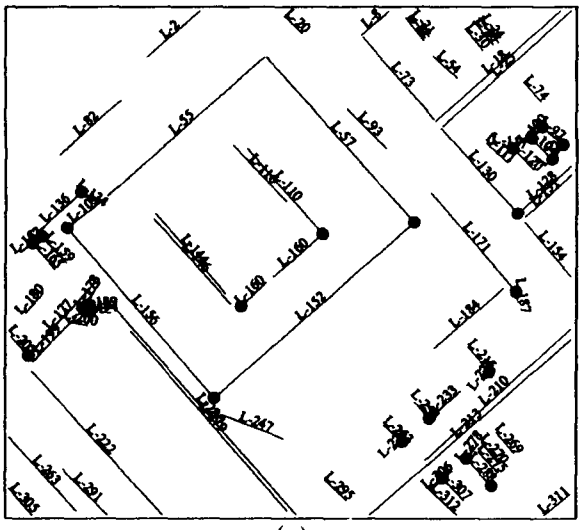


(a)



(b)

Figure 15: Vertex and line matches with vertex labels. (a) Right image. (b) Left image.



(a)



(b)

Figure 16: Vertex and line matches with line labels. (a) Right image. (b) Left image.

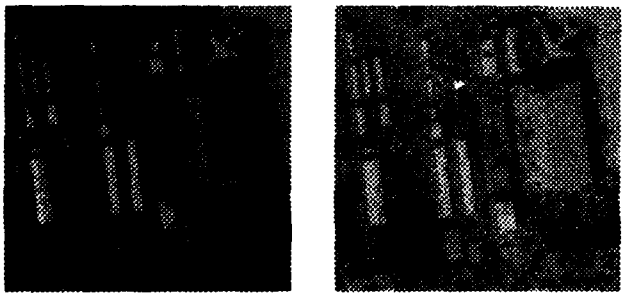


Figure 17: Right view and left view of a portion of LAX airport.

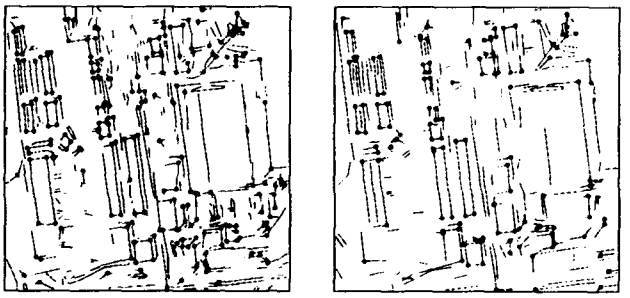


Figure 18: Lines and vertices.

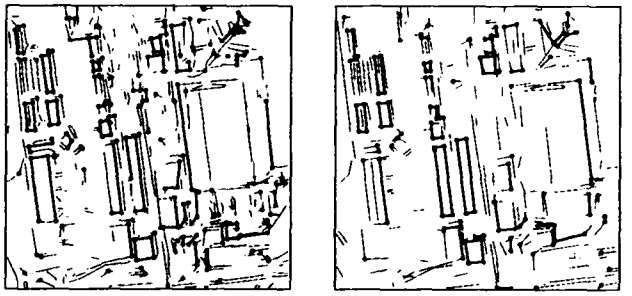


Figure 19: Edges (dark lines).

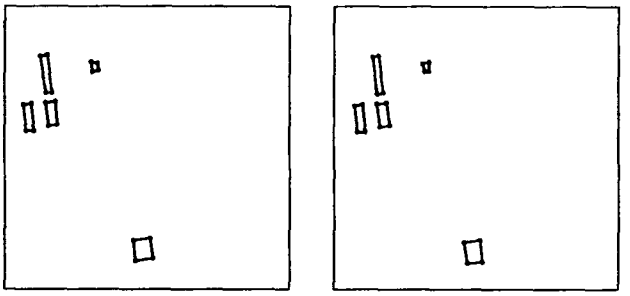


Figure 20: After matching at surface level.

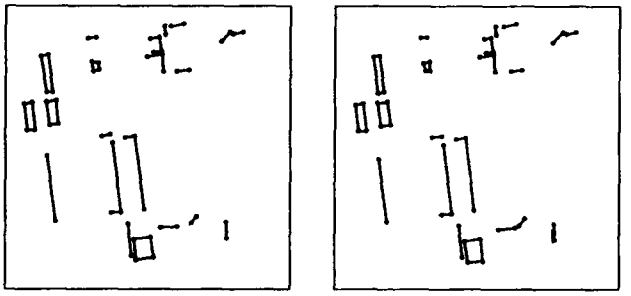


Figure 21: Final matches at edge level.

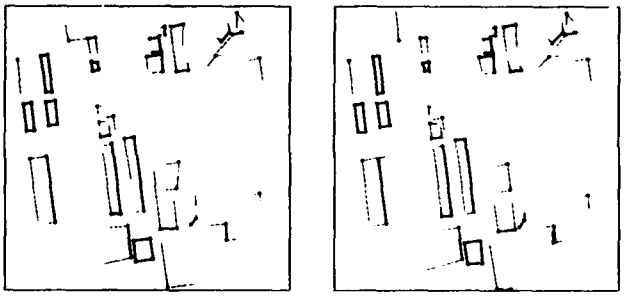


Figure 22: After matching at vertex level.

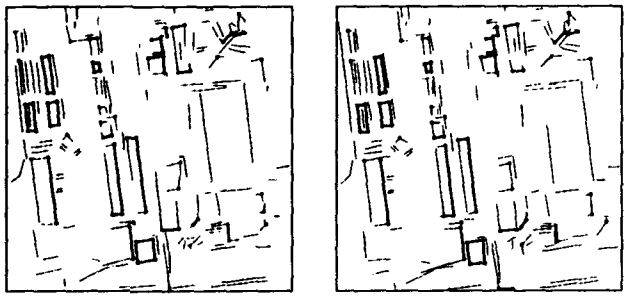
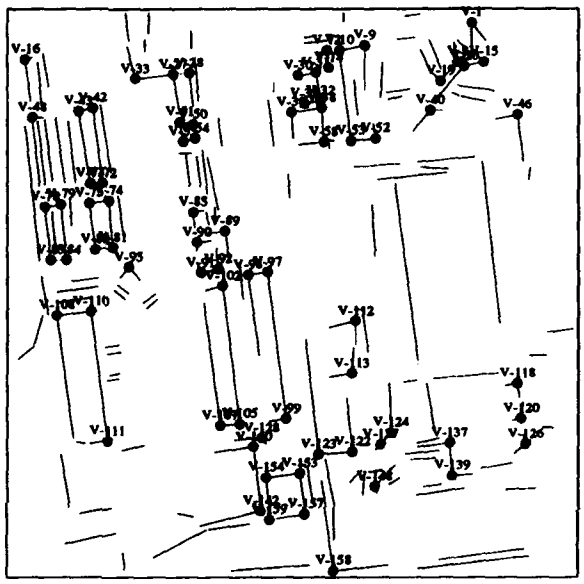
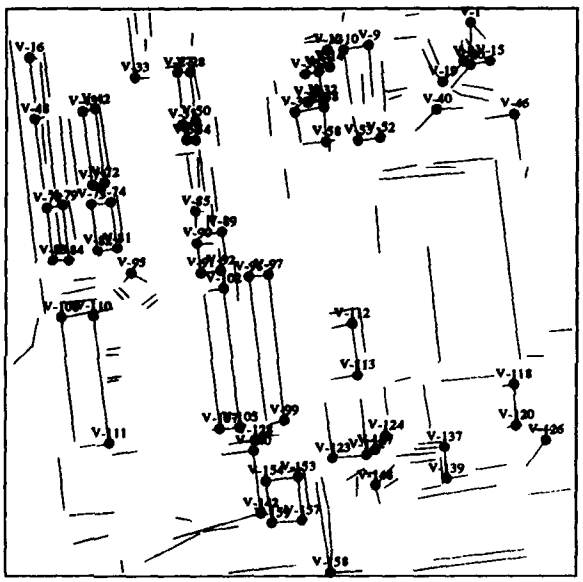


Figure 23: Final matches at line level.

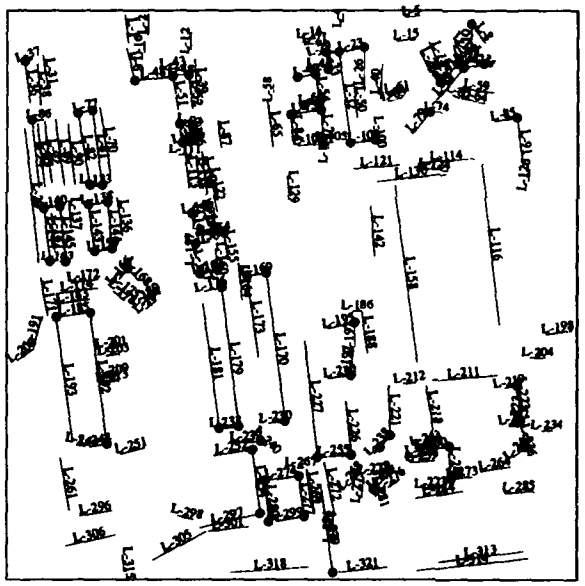


(a)

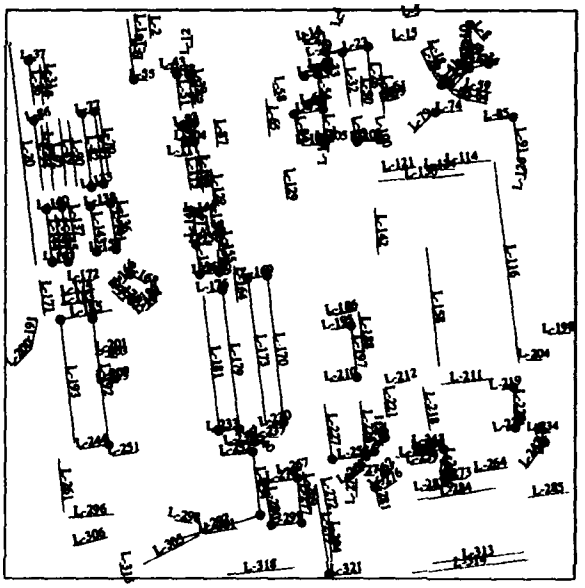


(b)

Figure 24: Vertex and line matches with vertex labels. (a) Right image. (b) Left image.



(a)



(b)

Figure 25: Vertex and line matches with line labels. (a) Right image. (b) Left image.

ON DIFFUSE REFLECTION AND PHOTOMETRIC STEREO

Lawrence B. Wolff
Computer Science Department
The Johns Hopkins University
Baltimore, Maryland 21218

Abstract

Diffuse reflection caused by subsurface multiple scattering from inhomogeneous dielectric materials is a very important phenomenon in computer vision. Such materials are quite prevalent and usually most of the sensed reflecting area from inhomogeneous dielectric materials results from diffuse reflection. Analysis and experimentation involving diffuse reflection in computer vision has typically assumed that it is Lambertian in nature. Non-Lambertian reflectance has previously implied the presence of specular reflection from a smooth or rough surface interface, and this has been recently studied in some detail even though the diffuse reflection component has still been assumed to be Lambertian. In fact we show that diffuse reflection from many common types of inhomogeneous dielectrics not only can significantly deviate from Lambertian behavior but that the shape of the angular distribution for diffuse reflection can be quite diverse, dependent upon material and surface parameters. In this paper we give serious consideration to the physical analysis of multiple subsurface scattering and propose diffuse reflection models based upon both radiative transfer theory and geometric optics.

As the behavior of the reflectance map for diffuse reflection with respect to surface orientation can be quite variable for different surface materials and roughnesses this can present a problem for determining accurate 3-D shape from current shape-from-shading and photometric stereo methodologies which require accurate knowledge of the reflectance map. We propose a novel technique for determination of 3-D shape from multiple point light source illumination without requiring knowledge of the diffuse reflectance map. This technique is based upon a *photometric partitioning* scheme which assumes that diffuse reflectance from any surface point is monotonic with respect to angle of incidence between the light source vector and the surface normal. The monotonicity assumption for diffuse reflection allows determination of the relative size of angles of incidence from multiple known incident source orientations from comparison of the relative magnitude of respective diffuse reflected values, regardless of how the unknown diffuse reflectance map may vary from point to point. Depending upon how many light sources are used the surface orientation for a normal can be constrained to lie within a partition of orientation space with decreasing size as the number of light sources gets larger. We suggest a method for how this scheme can be extended to simultaneously determine shape and the diffuse reflectance map.

1 INTRODUCTION

In the past decade the computer vision community has become increasingly aware of accurate modeling of the reflectance properties of materials for extraction of visual features. The works by Torrance and Sparrow [Torrance and Sparrow, 1967], and, Beckmann and Spizzichino [Beckmann and Spizzichino, 1963] have been amongst the most popular in providing vision researchers with an accurate modeling of the specular component of reflection from rough materials [Healey, 1987], [Wolff, 1987], [Tagare and deFigueiredo, 1989], [Nayar *et al.*, 1990]. While elaborate specular reflection models have been incorporated into computer vision methods, the diffuse component of reflection is still largely considered to be Lambertian. An exception is the paper by Tagare and deFigueiredo [Tagare and deFigueiredo, 1989] who propose as part of their m-lobed reflectance model for machine vision a functional approximation to a generalized Lambertian diffuse intensity distribution. Apart from analysis of reflected intensity distributions for diffuse reflection, Shafer [Shafer, 1985] proposed a color reflectance model, and Wolff [Wolff, 1991b] proposed a polarization reflectance model involving the diffuse component for inhomogeneous dielectrics.

The study of the precise nature of diffuse reflection from inhomogeneous dielectrics is very important for computer vision problems. For one, inhomogeneous dielectrics such as plastics, paints, glasses, ceramics, rubber, etc. are quite prevalent. Given non-extended light source illumination, diffuse reflection from these materials is far more abundant compared to the visible image area over which specular reflection occurs. From the analysis given in this paper, computer vision methods that rely on the assumption that the intensity distribution for diffuse reflection is Lambertian are likely to not be applicable to many of these common materials.

Our analysis of diffuse reflection relies on the theory of radiative transfer developed by Chandrasekhar [Chandrasekhar, 1960] for multiple scattering of incident light upon stellar and planetary atmospheres. The important problem of diffuse reflection and transmission from plane parallel atmospheres in astrophysics has a num-

ber of similarities with diffuse reflection from inhomogeneous dielectric materials. Incident light strikes gaseous molecules within an atmosphere whereupon some of the light is absorbed, and some of the light is scattered with an assumed intensity distribution respective to each individual molecule. Similarly, particles forming discontinuities in refractive index within a dielectric absorb and scatter light that has penetrated the surface boundary, and this process can be quantified using radiative transfer theory. In this paper, we propose diffuse reflection models for dielectric materials combining radiative transfer theory for multiple scattering with air-dielectric boundary surface effects. Radiative transfer theory has already been applied to analyzing diffuse reflection and transmission from colloidal suspensions [Reichman, 1973], [Orchard, 1969].

Subsequent to the analysis of diffuse reflection, we concentrate on the general application of photometric stereo to diffuse reflecting surfaces. Photometric stereo was originally pioneered by Robert Woodham [Woodham, 1978]. As originally proposed photometric stereo obtains local surface orientation using multiple light source incident orientations, assuming accurate knowledge of the reflectance map for the surface. Since then there has been considerable study and experimentation with this method, primarily for Lambertian diffusing surfaces [Silver, 1978], [Bruckstein and Kelley, 1983], [Onn and Bruckstein, 1990], [Kozera, 1991]. There has also been study of photometric stereo applied to specular surfaces [Ikeuchi, 1981], and surfaces that possess both a diffuse and specular component [Coleman and Jain, 1982], [Nayar *et al.*, 1988], but again, the diffuse component of reflection is assumed to be Lambertian in these cases. Tagare and deFigueiredo [Tagare and deFigueiredo, 1989] propose a theory of photometric stereo applied to a general class of reflectance maps with respect to uniqueness of solution for surface orientation and completeness of shape reconstruction. A functional approximation to the diffuse component is used based upon the final result of [Chandrasekhar, 1960]. This result however does not take into account the variety of material surface boundary effects that can strongly influence the nature of the diffuse reflection component.

Experimentation with photometric stereo on diffuse reflecting surfaces has been primarily performed on surfaces coated with non-glossy bright white paint [Silver, 1978]. No doubt white paint composed of titanium oxide or barium sulfate is the closest to being a perfect diffuser. In general it is recommended in [Silver, 1978] to build up a table relating surface orientation to quantities constructed from multiple reflected intensity values, each intensity value respective to a light source incident on a calibration object with similar reflectance properties. For 3 light sources, the following ratios would be constructed from reflected intensity values:

$$\frac{I_1}{I_1+I_2+I_3}, \quad \frac{I_2}{I_1+I_2+I_3}, \quad \frac{I_1}{I_1+I_2+I_3},$$

in order to cancel out diffuse "albedo" which represents a scaling factor for maximum reflected diffuse intensity (i.e., apparent grayness). The physical analysis in this paper shows in fact that the reflected intensity distributions of diffusers with varying "albedo" are not always scalar multiples of one another. The term "albedo" as commonly used in computer vision to mean scalar multiples of reflection component functions actually turns out to be simplistic in the case of diffuse reflection. The physical parameter that controls how much light is absorbed by a diffusing inhomogeneous dielectric surface is called the *single scattering albedo* which is appropriately defined to be the proportion of light energy that is singly scattered from an individual particle inhomogeneity, to the amount of light energy that was originally incident on the particle. A scattering albedo of 1.0 implies *conservative scattering* where all light is scattered and no energy is absorbed by the particle. We will see that the case of diffuse reflection produced from isotropic conservative multiple scattering is very close to being Lambertian. However, the shape of the reflected intensity distribution for diffuse reflection from nonconservative isotropic multiple scattering changes significantly with scattering albedos varying between 0.0 and 1.0. The point being made is that the shape of the reflected intensity distribution (i.e., reflectance property) for diffuse reflecting surfaces is frequently not uniform from point to point, and this is unfortunately the case for a surface painted with diffuse paints of varying grayness. The reflectance distribution at grayer points is not simply a fractional multiple of the nearly Lambertian distribution in the conservative scattering case. This poses a potential practical problem for current photometric stereo methodologies on diffuse reflecting surfaces that require accurate knowledge or make assumptions about the diffuse reflectance map. How can it be determined what reflectance map to use at each individual point ?

Presented in this paper is a new methodology for photometric stereo that we call *photometric partitioning* which obviates any need for precise knowledge about the diffuse reflectance map from point to point. This methodology only relies on the following 2 assumptions about the behavior of reflectance at a surface point: (i) diffuse reflectance is monotonic with respect to the angle of incidence formed by light source incident orientation and the surface normal; (ii) diffuse reflection is isotropic with respect to azimuth about the surface normal. Other than these 2 conditions for diffuse reflection, our method computes surface orientation equally well regardless of the values of the diffuse reflectance map and how this reflectance map may vary from point to point. We show experimentation with reconstructing the shape of part of a smooth ceramic vase with 3 light source photometric partitioning. We then show how the photometric partitioning method can be used to obtain simultaneous recovery of shape and the diffuse reflectance map.

2 THE PHYSICS OF DIFFUSE SCATTERING

The diffuse scattering of light as a natural phenomenon is quite ubiquitous. The color of a blue sky, an orange-red sunset, a blue-green sea, and, the color of inhomogeneous dielectric materials ranging from precious gems

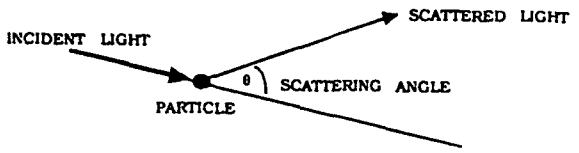


Figure 1

to ordinary clay can all be explained using the physics of diffuse scattering. Of interest to us is the predicted intensity distribution of diffusely scattered light from inhomogeneous dielectrics. All of this proceeds axiomatically from the *equation of transfer* for multiple scattering [Chandrasekhar, 1960].

The fundamentals of single scattering theory were established by Lord Rayleigh for particles smaller than the wavelength of incident light and by Mie for spherical particles of arbitrary size [Hulst, 1957], [Kerker, 1969]. We assume that the scattered light produced by a single small particle has an axially symmetric radiance distribution about the incident direction of light. The angular distribution of the scattered radiation is described by a *phase function* $P(cos\theta)$ where θ is the scattering angle of deflection away from the orientation of the incident light. See Figure 1. To be consistent with vision radiometric nomenclature [Horn and Sjoberg, 1979], the phase function describes the proportion of incident radiance that is scattered into a given direction. The phase function is defined so that if L is the radiance of light incident on a particle then $L \times P(cos\theta)/4\pi$ is the radiance scattered into a given direction forming an angle θ with the incident direction. The total irradiance of light that is scattered is given by:

$$\int_{unit\ sphere} L \times P(cos\theta)\frac{d\omega}{4\pi} .$$

If the phase function is constant for all angles θ then the scattering is said to be *isotropic*, otherwise it is *anisotropic*. In order for scattering to be conservative we must have:

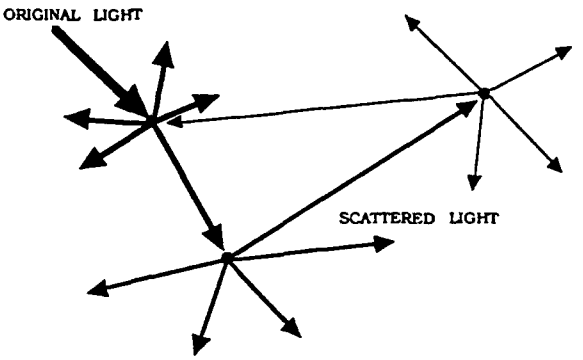
$$\int_{unit\ sphere} P(cos\theta)\frac{d\omega}{4\pi} = 1 .$$

In general

$$\int_{unit\ sphere} P(cos\theta)\frac{d\omega}{4\pi} = \sigma \leq 1 ,$$

and σ is referred to as the *albedo for single scattering* or simply *single scattering albedo*. In nonconservative cases when $\sigma < 1$ energy is absorbed by the particle. The factor of 4π is convenient so that for isotropic scattering with scattering albedo σ, $P(cos\theta) = \sigma$.

The scattering albedo is commonly dependent on the incident wavelength of light. The fact that the sky is blue is evidence that the scattering albedo for blue light in our atmosphere is significantly larger than for other wavelengths. The bright colors of rubyes and emeralds and the colors of ceramics is also due to the strong dependence of scattering albedo on wavelength. Reflected



SCATTERING AND INTER-SCATTERING OF LIGHT BY 3 PARTICLES

Figure 2

color that is sensed from objects illuminated with white light attests to the fact that nonconservative absorption is at work. Perfectly conservative materials should appear completely white when illuminated with this type of light.

Figure 2 depicts light scattering amongst multiple particles. In this case the radiance distribution of light that is incident upon a particle depends upon the past history of how this light was created from the previous scattering by other particles, which in turn depend upon the scattering of light by other particles before that, and so on. Figure 2 shows just how complicated the situation gets for only 3 particles, nothing compared to the actual scattering of light that occurs amongst thousands or millions of particles within an atmosphere or inhomogeneous materials. We assume here that the inhomogeneous medium being discussed, whether it be a gaseous atmosphere or a material, is *plane parallel* meaning that the medium can be subdivided into a set of parallel planes such that one of these planes represents the boundary of the medium, and along each individual plane the optical properties (e.g., particle density) is uniform. In this section surface effects at the boundary plane, such as refraction, are not considered. They are discussed in the next section.

Ultimately we want to derive an equation involving the radiance $I(\tau, \mu, \mu_0)$ as a function of optical depth from the boundary, τ, and incident and emergent directional cosines μ and μ_0 of light radiation with respect to the boundary normal. We can then evaluate this function at the boundary plane, namely, $I(\tau = 0, \mu, \mu_0)$, to get the proper law of diffuse reflection from multiple scattering. Because there is no dependence of I upon an asimuth parameter we are assuming an axially symmetric solution for the radiance about the boundary normal. The equation of transfer admits such a solution only if the single scattering phase function is isotropic. The fact that a number of common types of inhomogeneous dielectric materials appear equally bright regardless of rotation about the surface normal supports the physical isotropy for single scattering in such materials. The equation of transfer equates the rate of change of the radiance function $I(\tau, \mu, \mu_0)$ with respect to distance into the medium, to the sum of light radiation from 3 physical processes:

- 1. Light that is intercepted by (i.e., incident on) particles in the medium.
- 2. Light that is scattered (i.e., reradiated) as a result of physical process 1.
- 3. Light from the incident beam that passes in-between particles and is never scattered.

We introduce a *mass absorption coefficient* k_ν dependent upon frequency ν, defined by the equation

$$dI_\nu = -k_\nu \rho I_\nu ds , \quad (1)$$

where ρ is the mass density per unit volume, and ds is the differential distance that the light traverses in the medium. The term k_ν is proportional to the number of scatterers that intercept light per unit mass of the medium at incident frequency ν. Equation 1 describes the attenuation of light energy as a result of being intercepted by particles while traveling distance ds in the medium. The solution to equation 1 for I_ν is a simple calculus problem:

$$I_\nu = L \times e^{-(k_\nu \rho)s} , \quad (2)$$

which represents the proportion of the original incident radiance L that passes in-between all particles after traversing a distance s in the medium.

The light energy that is intercepted by particles is either scattered or absorbed. From equation 1 the radiance incident on particles per differential unit length, ds, is

$$k_\nu \rho I_\nu .$$

The radiance of light scattered at s into a vector direction having directional cosine μ with respect to the boundary normal and asimuth ϕ is given by:

$$k_\nu \rho \frac{1}{4\pi} \int_{-1}^{1} \int_{0}^{2\pi} P(\mu, \phi; \mu', \phi') I_\nu(s, \mu', \phi') d\phi' d\mu' .$$

The integration is performed over all incident directions, μ', ϕ'. Again, the phase function, P, in general is only dependent upon the angle between the incident direction μ', ϕ' and the scattering direction, μ, ϕ. Assuming isotropic scattering with single scattering albedo, σ, the phase function is

$$P(\mu, \phi; \mu', \phi') = \sigma .$$

Therefore the light energy reradiated by scattering is

$$k_\nu \rho \frac{1}{2} \sigma \int_{-1}^{1} I_\nu(s, \mu') d\mu' . \quad (3)$$

The equation of transfer for isotropic scattering is a result of combining expressions 1, 3 and 2 as follows;

$$\frac{dI_\nu(s, \mu)}{ds} = -k_\nu \rho I_\nu(s, \mu) + k_\nu \rho \frac{1}{2} \sigma \int_{-1}^{1} I_\nu(s, \mu') d\mu' - k_\nu \rho L e^{-(k_\nu \rho)s} , \quad (4)$$

where the last term is obtained by differentiating expression 2. $I_\nu(s, \mu)$ expresses the radiance of light radiated in a direction with directional cosine μ with the boundary normal, as a result of light incidence and multiple scattering.

We can eliminate needing to know the parameters k_ν and ρ by transforming the above equation expressed in terms of s, into an equation in terms of τ, the optical depth:

$$\mu \frac{dI(\tau, \mu, \mu_0)}{d\tau} = -I(\tau, \mu, \mu_0) + \frac{1}{2} \sigma \int_{-1}^{1} I(\tau, \mu', \mu_0) d\phi' d\mu' - L e^{-\tau/\mu_0} , \quad (5)$$

where we have suppressed the frequency dependence, ν, and the last term of this equation expresses the rate of change of the attenuation of incident light at radiance L with directional cosine μ_0 relative to the boundary normal. A more detailed explanation of this is given in [Wolff, 1991a].

Equation 5 is the *equation of transfer*. Chandrasekhar [Chandrasekhar, 1960] solves this integro-differential equation with over 100 pages of theoretical development. A summary emphasising the salient parts of this analysis is given in [Wolff, 1991a]. The solution for $I(\tau, \mu, \mu_0)$ at $\tau = 0$ can be expressed in terms of the *Chandrasekhar H-function*. The reflected radiance according to the law of diffuse reflection is;

(*Law of Diffuse Reflection*)

$$I(0, \mu, \mu_0) = \frac{\sigma}{4\pi} L \frac{\mu_0}{\mu + \mu_0} H(\mu) H(\mu_0) , \quad (6)$$

where, again, μ_0 and μ are the directional cosines of incident and reflected light with the boundary normal, respectively, σ is the single scattering albedo, and L is the incident radiance. The Chandrasekhar H-functions are also dependent upon the single scattering albedo, σ. An nth order approximation to the Chandrasekhar H-function can be expressed;

$$H(\mu) = \frac{1}{\mu_1 \cdots \mu_n} \frac{\prod_{i=1}^{n} (\mu + \mu_i)}{\prod_{\alpha} (1 + \kappa_\alpha \mu)} ,$$

defined in terms of the positive zeros, μ_i, of the $2nth$ Legendre polynomial, $P_{2n}(\mu)$, and the positive nonvanishing roots, κ_α, of the associated *characteristic equation*;

$$1 = \sum_{j=1}^{n} \frac{a_j \sigma}{1 - \kappa^2 \mu_j^2} .$$

[Wolff, 1991a] describes more details of the numerical evaluation of the Chandrasekhar H-function.

3 DIFFUSE REFLECTION IN COMPUTER VISION

We analyse the Law of Diffuse Reflection stated by equation 6 in the last section with respect to its application to phenomena important to computer vision. Under the

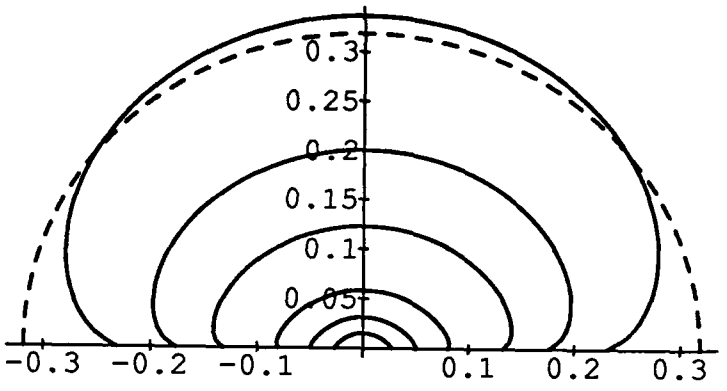


Figure 3a

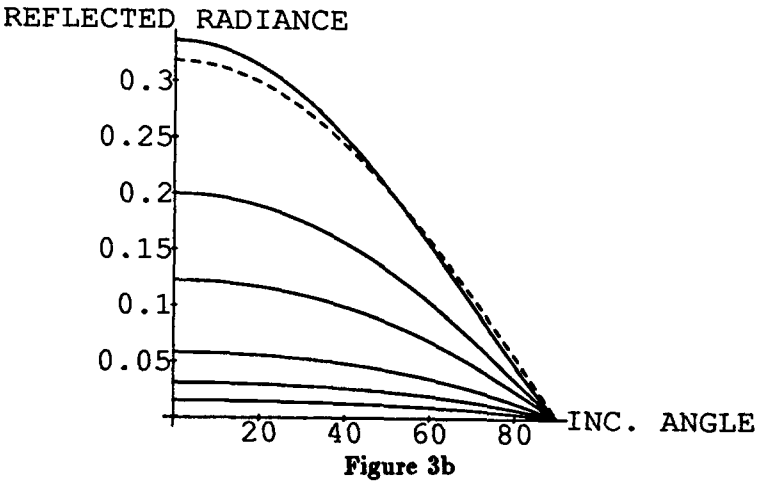


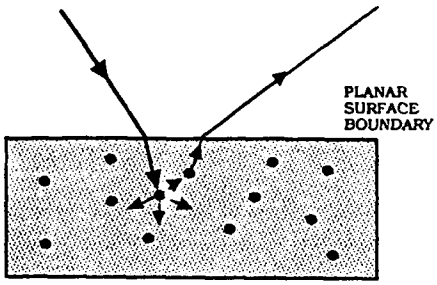
Figure 3b

assumption that particle inhomogeneities within a dielectric isotropically scatter incident visible light, equation 6 describes the radiance of diffuse subscattering just beneath the surface boundary with air. As discussed in the last section, for many diffuse reflecting materials this is strongly supported by their isotropic reflectance properties with respect to azimuth about the surface normal. The equation of transfer only admits axially symmetric solutions for an isotropic scattering phase function, $P(cos\theta)$ [Chandrasekhar, 1960]. In inhomogeneous dielectrics there might be different particle type inhomogeneities with various single scattering albedos. We assume that we can approximate the behavior of diffuse reflection from such materials by using the Law of Diffuse Reflection for a single scattering albedo represented by the optical averaging over many particles. Also, the single scattering albedo can be wavelength dependent in which case the Law of Diffuse Reflection is summed over all relevant wavelengths.

Equation 6 can be directly applied to rough diffusing inhomogeneous dielectric surfaces where surface detail smaller than the wavelength of incident light isotropically scatters this light to reasonable approximation. Many diffuse paints create this type of surface roughness upon application. Figure 3a depicts a polar scattering diagram for normal incident light with unit radiance for various single scattering albedos ranging from $\sigma = 1.0$ to $\sigma = 0.3$ according to the Law of Diffuse Reflection. For each point on the polar plot, the distance from the origin represents the radiance of reflected radiation scattered in the angular direction relative to the surface normal depicted by its angle relative to the vertical axis. The dashed plot depicts ideal conservative Lambertian reflectance. For conservative isotropic scattering the predicted Law of Diffuse Reflection from multiple scattering is indeed very close to being Lambertian. The radiance scattered parallel to normal is slightly larger than radiance scattered at oblique angles. Because scattering is conservative, the total reflected irradiance for the Law of Diffuse Reflection for $\sigma = 1.0$ is equal to the total incident irradiance, just as for the case of conservative ideal Lambertian reflectance. Notice the vast reduction in reflected radiance for all scattering angles for single scattering albedo decreasing from $\sigma = 1.0$ to $\sigma = 0.9$. At normal scattering the reflected radiance for $\sigma = 0.9$ is about a third of the reflected radiance of that for $\sigma = 1.0$. Because of multiple scattering it should be clear that the reflected radiance should degrade nonlinearly with respect to decreasing σ. The geometric progression of the proportion of light that is absorbed for a given single scattering albedo with respect to multiple scattering is extremely complicated to analyse combinatorially. The complete description lies in the equation of transfer, equation 5.

Looking more closely note on the polar plot for $\sigma = 0.9$ that the radiance scattered normal to the surface is actually slightly smaller than for the radiance scattered for most of the oblique angles. For decreasing σ going towards $\sigma = 0$ radiance scattered near grasing to the surface becomes more than twice as much as the radiance scattered normal to the surface. The shapes for the scattering polar plots are significantly different over the range of single scattering albedos. Figure 3b shows the graph of reflected radiance parallel to the surface normal versus angular orientation (in degrees) of incident light of unit radiance, for the same range of single scattering albedos as in Figure 3a. The dashed plot shows the ideal conservative Lambertian "$cos\theta$" Law. Again, for $\sigma = 1.0$, the Law of Diffuse Reflection is very nearly Lambertian. As σ decreases the darkening law for reflected light deviates more and more from Lambertian behavior, falling off more slowly as a function of angle of incidence.

Figures 3a and 3b for the Law of Diffuse Reflection imply that for diffuse reflecting materials with diffuse surface scattering, that points with different levels of "grayness" can have significantly different reflectance distribution properties. Defining diffuse "albedo" in the conventional sense as a scaling factor for a surface-wide uniform reflectance property is not physically precise. For instance, while the reflected radiance has decreased by a factor of 22 going from $\sigma = 1.0$ to $\sigma = 0.3$ for reflection normal to the surface with light at normal incidence, for reflection grasing the surface with light at normal incidence the reflected radiance has only decreased by a factor of 8. The single scattering albedo, σ, is the true physical parameter defining diffuse reflection, and variation of this physical parameter can produce significantly different reflectance properties from point to point on the same surface.



GEOMETRIC REFRACTION OF INCIDENT AND SCATTERED LIGHT BY A PLANAR SURFACE BOUNDARY

Figure 4

There are a number of diffuse reflecting surfaces which have an optically smooth surface boundary. Glossy paints possess diffuse reflecting properties on top of which there is a glossy sheen produced by specular reflection from an optically smooth surface layer. Many plastics and ceramics have this same characteristic. The effect of geometric refraction and Fresnel attenuation produced by optically smooth surface boundaries alters the Law of Diffuse of Reflection for diffusing materials having this type of surface layer. Figure 4 shows the geometric refraction of light incident on a smooth surface boundary, passing through the boundary and then subject to subsurface multiple scattering by particle inhomogeneities. Light that was originally incident at directional cosine μ_0 with respect to the surface normal, is now incident for subsurface diffuse scattering at directional cosine, $\overline{\mu_0}$, where according to Snell's Law [Siegal and Howell, 1981]:

$$\frac{sin(cos^{-1}\mu_0)}{sin(cos^{-1}\overline{\mu_0})} = n\,, \qquad (7)$$

where n is the simple index of refraction of the dielectric boundary. Accordingly,

$$\overline{\mu_0} = cos[sin^{-1}(\frac{sin(cos^{-1}\mu_0)}{n})]\,.$$

As depicted in Figure 4, the diffusely reflected light from multiple scattering is then geometrically refracted once again upon transmission back into air. The light that gets refracted into air at directional cosine, μ, was according to Snell's Law originally diffusely reflected below the boundary surface at directional cosine $\overline{\mu}$, where;

$$\overline{\mu} = cos[sin^{-1}(sin(cos^{-1}\mu) \times n)]\,.$$

Upon reflection at a planar surface boundary light becomes attenuated according to the Fresnel reflection coefficients, $F(n,\mu)$, for different components of polarisation, as a function of index of refraction, n, and directional cosine, μ [Siegal and Howell, 1981]. We assume here that incident light is unpolarised. For incident light with radiance L, light that is specularly reflected on the opposite side of the normal at the same angle μ has radiance $F(n,\mu) \times L$. The energy transmitted into the material is therefore proportionally $1 - F(n,\mu)$. For transmission of light from the dielectric into air, since light is going from a medium with index of refraction, n, to

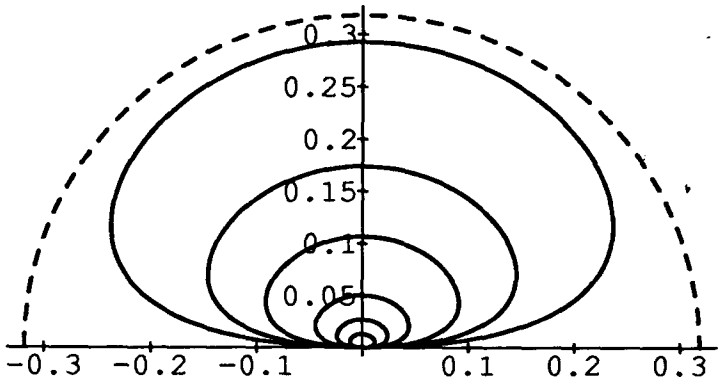


Figure 5a

index of refraction, 1.0, the attenuation for specular reflection is now $F(1/n,\mu)$ for incident directional cosine, μ. The proportion of energy transmitted into air is therefore $1 - F(1/n,\mu)$ [1]. To first order approximation the Law of Diffuse Reflection for inhomogeneous dielectrics with optically smooth surface boundaries is:

(*Optically Smooth Surfaces*)

$$[1-F(n,\mu_0)] \times \frac{\sigma}{4\pi} L \frac{\overline{\mu_0}}{\overline{\mu}+\overline{\mu_0}} H(\overline{\mu}) H(\overline{\mu_0}) \times [1-F(1/n,\overline{\mu})]\,. \qquad (8)$$

The second order approximation accounts for light that is reflected from the surface-air boundary back into the dielectric, and then diffusely scattered again. As there is *total internal reflection* for scattering angles greater than the *critical angle*[2] $sin^{-1}(1/n)$ (about 36° for n = 1.7 for most plastics and ceramics) the amount of this light reflected back into the dielectric is significant. The third order approximation which accounts for light that is again reflected back into the dielectric from a second diffuse scattering is not significantly different from the second order approximation.

Figures 5a and 5b plot the second order approximation to the Law of Diffuse Reflection for diffusers with optically smooth boundary surfaces for n = 1.7 typical of a number of common dielectrics such as ceramic and plastic. The plots are defined in the same way as for Figures 3a and 3b using the same single scattering

[1] Due to a light wave *impedance* correction between media with different indices of refraction, the proportion of radiance, L, measured in air, that is transmitted from a medium of index of refraction, 1.0, to a dielectric medium of index of refraction, n, is $n \times (1 - F(n,\mu)) \times L$ as measured in the dielectric. Similarly, the proportion of radiance, L, measured in the dielectric with index of refraction, n, that is transmitted into air with index of refraction, 1.0 is $(1/n) \times (1 - F(1/n,\mu)) \times L$, as measured in air. However, since we are measuring incident light radiance in air and reflected radiance also in air, the impedance correction terms cancel out.

[2] This *critical angle* is determined by Snell's Law, equation 7, where $\overline{\mu_0}$ now represents the cosine of the internal angle of incidence within the dielectric and μ_0 represents the cosine of the transmitted angle. Solving for the critical angle, $\overline{\mu_0}$, when the transmitted angle, μ_0, is 90° (i.e., grasing), gives $sin^{-1}(1/n)$.

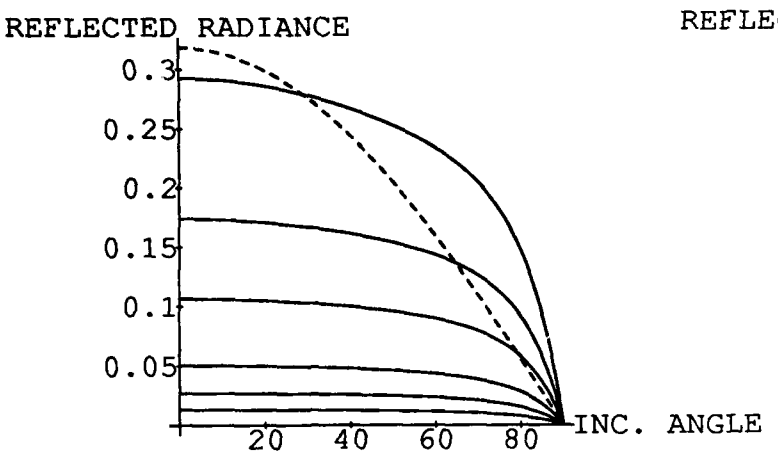


Figure 5b

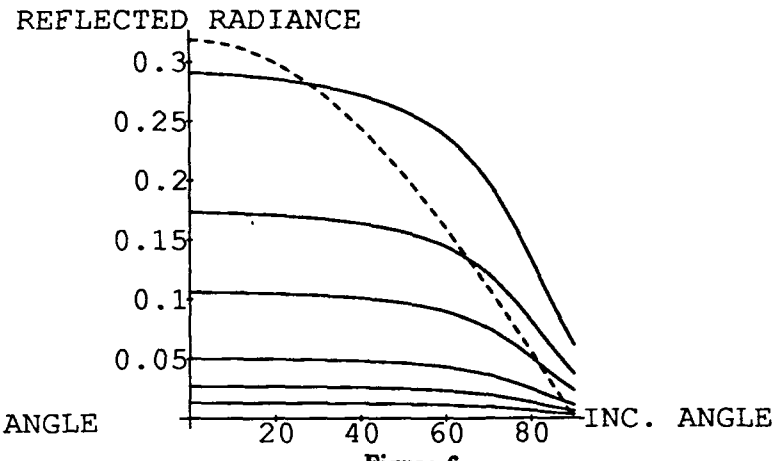


Figure 6

albedos σ. For $\sigma = 1.0$, the total diffuse reflected irradiance is less than the total incident irradiance since part of the reflected light energy is contributed to a specular component. Interestingly, the reflectance radiance distributions for different scattering albedos, σ, for an optically smooth surface boundary are more similar than they were for a diffusing surface boundary. However, the plots in Figures 5a and 5b are more deviant from ideal Lambertian behavior. Figure 5a shows that almost no light is scattered in directions close to grasing the surface. Light that transmits through the boundary surface layer after subsurface diffuse scattering only results from diffusely scattered light less than or equal to the critical angle. Near the critical angle of internal incidence transmitted light nearly grases the surface at 90° to the boundary normal. Also, near this angle almost all light is internally reflected and very little light is transmitted explaining why the polar plots in Figure 5a depict little or no diffuse reflection almost orthogonal to the boundary normal.

Figure 5b shows the strong deviation of diffuse reflection from the Lambertian "$cos\theta$" Law for all scattering albedos, σ. The falling off of reflected radiance as a function of angle of incidence, starting from normal light incidence, is much less than predicted for Lambertian behavior. Near grasing incidence the reflected radiance for a diffuser with optically smooth boundary, starts falling off rapidly going to sero at 90°. Figure 6 shows the Law of Diffuse Reflection for a rough surface boundary composed of a Gaussian distribution of optically planar microfacets with respect to orientation, with mean orientation the surface normal and a standard deviation of 10°. Not only does the reflected radiance as a function of angle of incidence get even flatter, but at grasing incidence the reflected radiance is non-sero. This is easy to explain since grasing incidence produces non-sero transmission into the surface through planar microfacets oriented oblique to the surface normal. Such transmission is then diffusely scattered and transmitted back out into air. Not surprisingly the larger the standard deviation of the orientation roughness distribution for the planar microfacets, the flatter is the law of darkening as a function of angle of incidence and the larger the diffuse reflected radiance at grasing incidence.

4 THE METHOD OF PHOTOMETRIC PARTITIONING

4.1 SHAPE RECOVERY AND THE MONOTONICITY ASSUMPTION FOR DIFFUSE REFLECTION

The discussion in section 3 about the nature of diffuse reflection from typical surfaces encountered in computer vision illustrates not only how diffuse reflection can deviate from Lambertian behavior, but how diverse this behavior can be according to the optical properties of the surface boundary and the single scattering albedo. Furthermore, these optical properties commonly vary between surface points producing in turn variable diffuse reflectance properties over the same surface. Unless these optical characteristics are precisely known from point to point, there can be a significant uncertainty in knowledge about the diffuse reflectance properties at each point. Methodologies requiring precise knowledge of the diffuse reflectance map for shape recovery may produce significantly inaccurate results for a number of commonly occuring surface materials. These include such methodologies as shape-from-shading [Horn, 1975] and photometric stereo [Woodham, 1978].

We investigate a novel methodology for the recovery of shape that utilizes multiple point light source illumination. In this sense our methodology is a variation of photometric stereo, but the main conceptual difference is that we do not require knowledge of the diffuse reflectance map. Such a methodology is generally applicable to the shape recovery of inhomogeneous dielectric surfaces regardless of the possible variety of diffuse reflectance discussed above. We will also see that, theoretically speaking, by adding more and more light sources our methodology can produce almost arbitrary accuracy in surface orientation measurement. This is certainly not the case for conventional photometric stereo without any knowledge of the reflectance map.

Our methodology relies upon an important assumption about diffuse reflection stated as follows;

MONOTONICITY ASSUMPTION FOR DIFFUSE REFLECTION: *Diffuse reflected radiance from inhomogeneous dielectrics resulting from subsurface scattering is monotonic with respect to the angle of incidence between*

the incident orientation of light and the surface normal. This is true for any emittance direction.

This assumption is robust for inhomogeneous dielectrics possessing isotropic scattering inhomogeneities and having a surface boundary that either scatters isotropically or according to Snell's Law and the Fresnel reflection coefficients, as supported by Figures 3b, 5b, and 6. It should be clear that monotonicity should consistently imply *monotonic decreasing* with respect to angle of incidence. The monotonicity assumption for diffuse reflection is also intuitively appealing. For light incident on a surface at radiance, L_i, and angle, θ_i, respective to the surface normal, the total incident irradiance is $L_i cos\theta_i$ [Horn and Sjoberg, 1979] and this is monotonic decreasing with respect to θ_i. Hence the total irradiance of light incident on a surface which is then subject to diffuse scattering decreases as a function of angle of incidence. This does not preclude some possibly pathological case of surface roughness where for some reason most of the planar microfacet surface normals are oriented extremely oblique to the surface normal and none or very few are actually oriented parallel to the surface normal. It is theoretically possible in this case for monotonicity of diffuse reflection to break down in some ranges of angle of incidence, but in such a case it is fair to call into question exactly what a surface normal physically means at a point. Nonetheless, as seen in section 3, the monotonicity assumption for diffuse reflection is by far more robust than the assumption about Lambertian reflectance.

Because of its directional nature, the specular component of reflection violates monotonicity for reflected radiance with respect to angle of incidence for both smooth and rough surfaces. Regions on inhomogeneous dielectric surfaces from which there is observed significant specular reflection must be identified in order to know where monotonicity is violated. This is actually very easy for dielectrics since as presented in [Wolff, 1989] specular reflecting regions on smooth and rough surfaces can be passively identified by placing a polarizing filter in front of a video camera. For photometric stereo once specular regions are identified they automatically provide surface orientation information about these regions because the incident orientation of the light source from which the specular reflection is produced is known (i.e., the surface normal bisects the light source and viewing directions).

Consider the mutual illumination area of 3 light sources (i.e., surface orientations that do not lie in shadow for any of the light sources individually turned on) as depicted in Figure 7 and assume that the specular regions produced by each one of the light sources has been identified. Surface points that are mutually illuminated and that are not in specular regions, diffusely reflect light from each one of the light sources individually into the camera sensor. At a given surface point, the diffuse reflected radiance values I_1, I_2 and I_3 respective to each light source are produced for the same emittance angle (i.e., angle between the surface normal and the viewing vector) but at possibly different angles of incidence. Given the monotonicity assumption for diffuse reflection, a simple comparison of the relative magnitudes of I_1, I_2 and I_3 defines a natural partitioning of

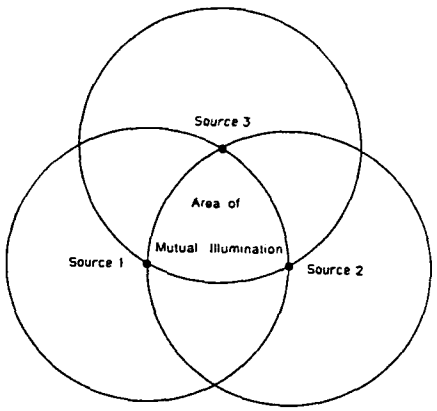
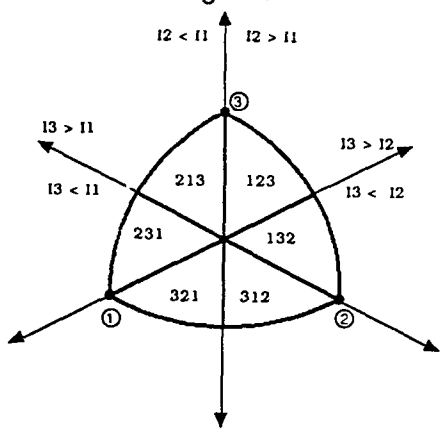


Figure 7

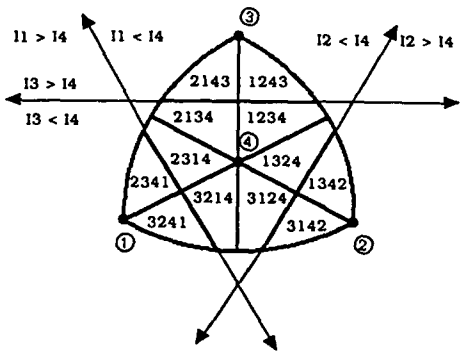


PHOTOMETRIC PARTITIONING OF MUTUALLY ILLUMINATED AREA (3 LIGHT SOURCES)

Figure 8

the mutual illumination area of surface orientations as depicted in Figure 8. The relative magnitudes of the diffuse reflected intensities are in the exact opposite order (due to monotonic *decreasing*) as the relative magnitudes of the respective angles of incidence made between each respective light source and the surface normal.

Consider the comparison of the angles of incidence for just 2 light sources. If the incident orientations for light sources 1 and 2 are represented by points on the unit Gaussian sphere, the loci of points on the Gauss sphere representing surface orientations that are simultaneously equi-angular with these two incident source orientations



PHOTOMETRIC PARTITIONING OF MUTUALLY ILLUMINATED AREA (4 LIGHT SOURCES)

Figure 9

is a great circle perpendicularly bisecting the arc between the 2 points for the two light sources. If the angle of incidence for a surface normal is smaller with light source 1 than for light source 2, then the surface normal must lie to the left of the equi-angular partition line between light sources, within the mutual illumination region as depicted in Figure 8. If the angle of incidence for a surface normal is larger with light source 1 than for light source 2, then the surface normal must lie to the right of the equi-angular partition line, within the mutual illumination region. Equality would place the surface normal on the partition line itself. The monotonicity assumption for diffuse reflection allows comparison of the relative magnitude of the angle of incidence between the surface normal and each one of the point light sources, without any knowledge of the diffuse reflectance map.

Figure 8 shows the 6 partition regions within the mutual illumination area created by equi-angular partition lines between 3 light sources. In this example the incident orientations of 3 light sources are mutually orthogonal so that the incident orientations for each one of the light sources comprises each of the corners of the mutual illumination area. The labeling numbers ABC for each of the regions in Figure 8 represent that the reflected diffuse intensities for the surface orientations in this region are in the order $A < B < C$ respective to light sources A, B and C. This implies that the angles of incidence are in the opposite order $A > B > C$ respective to light sources A, B and C. Figure 9 shows the 12 partition regions within the same mutual illumination area created by equi-angular partition lines between 4 light sources. The 4th light source is added to the center of the lighting configuration depicted in Figure 8. The same labeling scheme for partition regions holds.

For N light sources the total number of combinations of comparison outcomes that can be made between any 2 diffuse reflectance measurements is

$$P(N,2) = \frac{N!}{(N-2)!} = N(N-1).$$

This defines the maximum number of partitions within a mutual illumination area that can be created with N light sources. Figures 8 and 9 depict lighting configurations that achieve the maximum number of partitions for 3 and 4 light sources respectively. An example of suboptimal partitioning is a configuration of 3 coplanar light source orientations only defining 4 surface orientation partition regions.

We now consider the average angular error produced by partitioning of a mutual illumination surface orientation region the size of the octant of a sphere. This is the same size as the mutual illumination area depicted in Figures 8 and 9. Once we know which surface orientation partition region a surface normal belongs too, what orientation value should be assigned to the normal ? The best assignment of orientation value should minimise average angular error. Assuming initially that all surface orientations within the mutual illumination area are equally probable, assigning the centroid orientation with respect to the vertices of the respective partition region comes very close to minimizing overall average angular error.

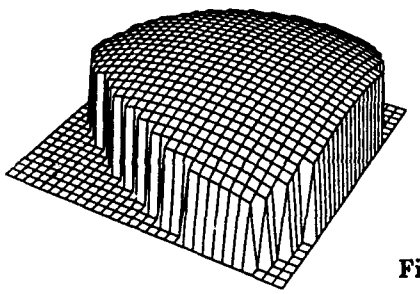


Figure 10a

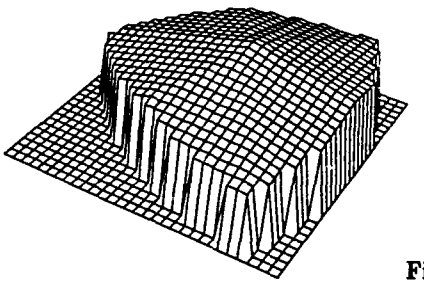


Figure 10b

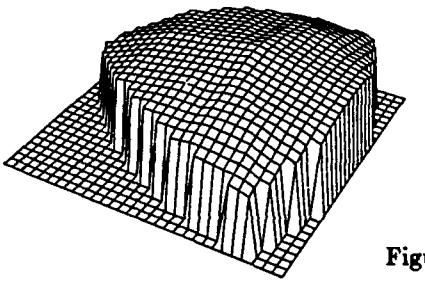


Figure 10c

For a diffuse reflecting sphere, assigning orientation values in this way to surface normals, for 3 light sources as in Figure 8 the average angular error is 12.1°, and for 4 light sources as in Figure 9 the average angular error is 8.7°. The portion of the sphere for which shape is being determined is shown in Figure 10a. Figure 10b shows a simulation of shape reconstruction using photometric partitioning with 3 light sources, and Figure 10c shows a simulation of shape reconstruction using photometric partitioning with 4 light sources (respective to the lighting configurations in Figures 8 and 9). The shapes in Figures 10b and 10c have the appearance of roughhewn unfinished sculptures of the sphere in Figure 10a. Using only 3 light sources, the method of photometric partitioning provides more rudimentary shape information, while photometric partitioning with additional light sources provides more shape precision. Theoretically, arbitrary precision of shape could be determined with more and more point light sources, without any knowledge of the diffuse reflectance map. Shape accuracy is also increased for a fixed number of light sources by decreasing the size of the mutual illumination area covered. If in Figures 8 and 9 the mutual illumination area were reduced to one-fourth of the octant of a sphere, the respective average angular errors would be reduced by one-half to 6.0° and 4.3°.

These results are a significant improvement over conventional photometric stereo assuming a Lambertian re-

flectance map. For an inhomogeneous dielectric with a smooth surface with diffuse reflectance dependent upon angle of incidence simulated in Figure 5b, an assumption of Lambertian reflectance would produce an average angular error of over 15° for all single scattering albedos. For an inhomogeneous dielectric with a rough surface with diffuse reflectance dependent upon angle of incidence simulated in Figure 6, an assumption of Lambertian reflectance would produce an average angular error of over 20° for all single scattering albedos. These are with respect to simulations for reconstruction of the octant of the sphere shown in Figure 10a.

Figure 11 shows an experimental result of shape determination of part of a smooth ceramic vase from photometric partitioning with 3 light sources configured orthogonal to one other. The square on the photograph depicts where the shape was reconstructed. Note how the shape reconstruction in Figure 11 is "smoother" than the reconstruction of the sphere in Figure 10b. This is due to additional surface orientations being assigned to pixels from the appropriate orientation midpoints of sides of partition regions in Figure 8 according to where 2 of the camera intensity values were equal within gray level repeatability (determined to be ± 2 gray levels). When all 3 gray values were equal to within repeatability, the orientation value is $(0, 0)$ in gradient coordinates.

The shape reconstruction in Figure 12 was obtained from the same camera data but assuming that the reflectance map is Lambertian with variable albedo scale factor. Note how this shape reconstruction is considerably flatter than it should be. This is consistent with the fact that diffuse reflectance from the smooth ceramic vase does not fall off as fast with respect to angle of incidence as for Lambertian reflectance (predicted in Figure 5b !). If Lambertian reflectance is assumed, photometric stereo believes that the relative angle of incidence between the surface normal and each of the light sources is not varying as much as it really is from point to point. Therefore, the surface normals appear not to vary nearly as much as they should be and the reconstructed surface is correspondingly flatter.

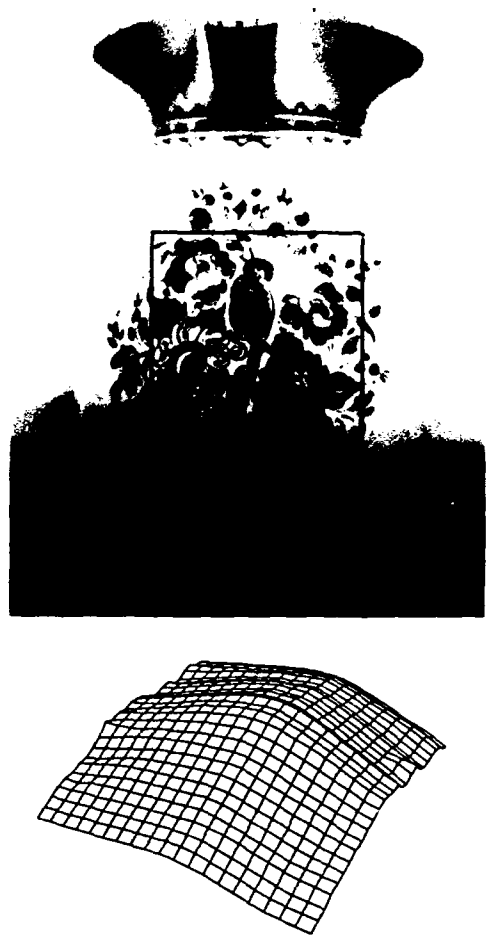


Figure 11

4.2 SIMULTANEOUS DETERMINATION OF SHAPE AND DIFFUSE REFLECTANCE

The method of photometric partitioning not only provides a methodology for recovering shape from unknown diffuse reflectance, but can actually be used to determine the diffuse reflectance map itself on a surface of unknown shape. Figure 13 depicts a lighting configuration that can be used for the simultaneous determination of 2-D surface orientation and the diffuse reflectance map from photometric partitioning. A total of 6 point light sources are used here over an incident 2-D orientation range of 180°. The solid lines represent equiangular partition lines for adjacent light sources. Determining the light source that produces the maximum diffuse reflected intensity with respect to all 6 light sources, constrains the normal at a surface point to be within an orientation range of 36° for normals oriented somewhere within the 4 middle solid partitions, and an orientation range of 18° for normals oriented more oblique than the extreme left and right solid partition lines (since normals cannot be more than 90° relative to camera viewing).

The jagged lines in Figure 13 represent equiangular partition lines for light sources that have one light source in between. This is except for the horizontal jagged lines which represent directions orthogonal to viewing. The jagged partition lines can be used to constrain all 2-D surface orientations to within 18°. Consider now comparing the diffuse reflected radiances from the 2 light sources on either side of the light source producing the maximum diffuse reflectance. Recall that normals whose

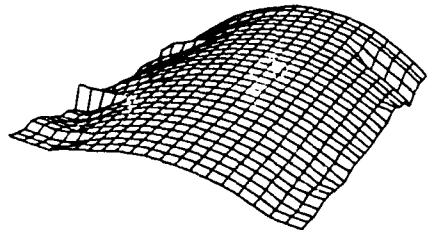


Figure 12

maximum diffuse reflectance is produced from either the left or right most extreme light sources are already constrained to within 18°, so we are now considering normals whose maximum diffuse reflectance is produced from one of the 4 middle light sources. Depending upon which diffuse reflected radiance is greater, a normal constrained within a solid partition can now be determined to lie on one side of the jagged partition line bisecting the solid partition. That is, the orientation for a normal can be constrained within 18°. Unfortunately due to symmetry no additional equiangular partition lines are produced from the 6 light sources.

In brief, using the monotonicity assumption and isotropy of diffuse reflectance, the orientation range of a normal at a surface point can be restricted within 18° using the configuration in Figure 13 by observing the light sources producing the largest and second largest diffuse reflected radiance. Assigning an orientation value that bisects these 18° partitions gives a worst case error of 9° and an average error of 4.5°, assuming all 2-D orientations are equally probable. In general using an even number, N, of point light sources to give room for the camera to look vertically down as in Figure 13, the average 2-D orientation error will be $180/(8 \times (N-1))$ degrees.

Once 2-D surface orientation is determined using photometric partitioning, the light sources in Figure 13 provide different diffuse reflectance measurements, sampling the diffuse reflectance map at up to 5 different known angles of incidence for each surface point. At each visible surface point the diffuse reflectance map can be sampled for at least 3 angles of incidence between 0° and 90°. The maximum number of 5 samplings at different angles of incidence occurs for points with normals within 36° of the camera viewing orientation, and not parallel to any of the solid or jagged equiangular partition lines. Clearly, the number of samplings can be increased with more available light sources and this may be important for surfaces with nonuniform diffuse reflectance properties from point to point. For a surface with nearly uniform diffuse reflectance from point to point and having a diversity of visible surface normal orientations, the diffuse reflectance map can be very finely sampled not only for angle of incidence, but for emittance angle as well.

The extension of the configuration in Figure 13 to 3-D surface orientations is a bit more tedious but conceptually simple. We will shortly report experimental results for diffuse reflection from different dielectrics using the configuration in Figure 13.

5 FUTURE WORK AND CONCLUSION

In this paper we have delved deeply into the physical nature of diffuse reflection from inhomogeneous dielectric surfaces and have found that the Lambertian description can be quite simplistic for a number of ordinary surfaces. Diffuse reflection is in fact a very rich area that has previously not been closely studied in computer vision. It can take on many different properties dependent upon

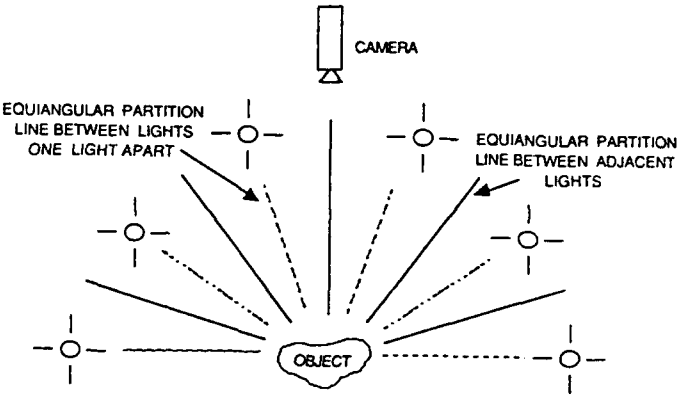


Figure 13

a number of optical parameters such as single scattering albedo and surface boundary roughness. We have studied diffuse reflection for the case of subsurface isotropic scattering and this is indeed applicable to a wide variety of materials.

There are surfaces for which subsurface anisotropic scattering applies (e.g., types of wood) and this topic is currently being studied and to be reported in the near future. Lord Rayleigh investigated the scattering of light from particles according to the dipole moment that the electric field of incident light induces on the particle. This is dependent upon the *polarizability*, α, of the particle. The *Rayleigh scattering* phase function is axially symmetric and given by the following proportionality:

$$P(cos\theta) \propto \alpha^2(1+cos^2\theta)\,.$$

Such an anisotropic scattering phase function is applicable to our own earth's atmosphere and is fairly accurate in accounting for the radiance distribution of transmitted sunlight across the sky.

The diversity of the nature of diffuse reflection should not be considered a hindrance to computer vision research. On the contrary, because of the prevalence of this phenomenon and its dependence upon important optical parameters it provides a great opportunity for utilising diffuse reflection as a tool for extracting a number of object features. This can include determination of surface roughness, and even material classification from measurement of the diffuse reflectance map to derive properties such as single scattering albedo of subsurface particle inhomogeneities. This requires the development of a new set of vision methodologies and opens up a whole new subarea of research.

For the many varieties of diffuse reflection properties that can exist we have identified an invariant, namely the monotonicity assumption that reflected diffuse radiance is monotonic with respect to angle of incidence. We utilised this monotonicity assumption to develop a new photometric stereo methodology for shape determination which is generally applicable to a wide variety of isotropic diffuse reflecting surfaces. We have also proposed using the photometric partitioning method for simultaneous determination of shape and the diffuse reflectance map. Because photometric partitioning deter-

mines shape without knowledge of the diffuse reflectance map, tabulating the diffusely reflected values for known computed normals derives the diffuse reflectance map from multiple known incident light orientations. We are currently performing experimentation with this technique.

Acknowledgment

This research was supported in part by an NSF Research Initiation Award, grant IRI-9111973.

References

[Beckmann and Spizzichino, 1963] P. Beckmann and A. Spizzichino. *The Scattering of Electromagnetic Waves from Rough Surfaces.* Macmillan, 1963.

[Bruckstein and Kelley, 1983] R. Ray J. Bruckstein and R.B. Kelley. Error analysis of surface normals determined by radiometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 5(6):631–645, 1983.

[Chandrasekhar, 1960] S. Chandrasekhar. *Radiative Transfer.* Dover Publications, New York, 1960.

[Coleman and Jain, 1982] Coleman and Jain. Obtaining 3d shape of textured and specular surfaces using four-source photometry. *Computer Vision Graphics and Image Processing (CVGIP)*, 18(4):309–328, 1982.

[Healey, 1987] G. Healey. Local shape from specularity. In *Proceedings of the IEEE first International Conference on Computer Vision (ICCV)*, pages 151–160, London, June 1987.

[Horn and Sjoberg, 1979] B.K.P. Horn and R.W. Sjoberg. Calculating the reflectance map. *Applied Optics*, 18(11):1770–1779, June 1979.

[Horn, 1975] B.K.P. Horn. Obtaining shape from shading information. *The Psychology of Computer Vision*, pages 115–155, 1975.

[Hulst, 1957] H.C. Van De Hulst. *Light Scattering by Small Particles.* John Wiley and Sons, New York, 1957.

[Ikeuchi, 1981] K. Ikeuchi. Determining surface orientations of specular surfaces by using the photometric stereo method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(6):661–669, November 1981.

[Kerker, 1969] M. Kerker. *The Scattering of Light and other Electromagnetic Radiation.* Academic Press, 1969.

[Kozera, 1991] R. Kozera. Existence and uniqueness in photometric stereo. *Applied Mathematics and Computation*, 44(1):1–104, 1991.

[Nayar *et al.*, 1988] S. Nayar, K. Ikeuchi, and T. Kanade. *Extracting Shape and Reflectance of Lambertian, Specular, and Hybrid Surfaces.* CMU Technical Report RI-TR-88-14, August 1988.

[Nayar *et al.*, 1990] S. Nayar, K. Ikeuchi, and T. Kanade. Surface reflection: Physical and geometrical perspectives. In *Proceedings of the DARPA Image Understanding Workshop*, pages 185–212, September 1990.

[Onn and Bruckstein, 1990] R. Onn and A. Bruckstein. Integrability disambiguates surface recovery in two-image photometric stereo. *International Journal of Computer Vision (IJCV)*, 5(1):105–113, 1990.

[Orchard, 1969] S. Orchard. Reflection and transmission of light by diffusing suspensions. *Journal of the Optical Society of America*, 59(12):1584–1597, December 1969.

[Reichman, 1973] J. Reichman. Determination of absorption and scattering coefficients for nonhomogeneous media 1: Theory. *Applied Optics*, 12(8):1811–1815, August 1973.

[Shafer, 1985] S. Shafer. Using color to separate reflection components. *Color Research and Application*, 10:210–218, 1985.

[Siegal and Howell, 1981] R. Siegal and J.R. Howell. *Thermal Radiation Heat Transfer.* McGraw-Hill, 1981.

[Silver, 1978] W.M. Silver. Determining shape and reflectance using multiple images. Master's thesis, MIT, June 1978.

[Tagare and deFigueiredo, 1989] H.D. Tagare and R.J.P. deFigueiredo. A theory of photometric stereo for a general class of reflectance maps. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 38–45, San Diego, June 1989.

[Torrance and Sparrow, 1967] K. Torrance and E. Sparrow. Theory for off-specular reflection from roughened surfaces. *Journal of the Optical Society of America*, 57:1105–1114, 1967.

[Wolff, 1987] L.B. Wolff. Spectral and polarization stereo methods using a single light source. In *Proceedings of the IEEE first International Conference on Computer Vision (ICCV)*, pages 708–715, London, June 1987.

[Wolff, 1989] L.B. Wolff. Using polarization to separate reflection components. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 363–369, San Diego, June 1989.

[Wolff, 1991a] L.B. Wolff. *On Diffuse Reflection.* Johns Hopkins Technical Report CS-91-18, October 1991.

[Wolff, 1991b] L.B. Wolff. *Polarization Methods in Computer Vision.* PhD thesis, Columbia University, January 1991.

[Woodham, 1978] R.J. Woodham. *Reflectance map techniques for analyzing surface defects in metal castings.* PhD thesis, MIT AI Lab Tech Report AI-TR-457, June 1978.

Subsection C:

Motion

Motion Tracking on the Spatiotemporal Surface

H. H. Baker **T. D. Garvey**

Artificial Intelligence Center, SRI International
333 Ravenswood Ave., Menlo Park, CA 94025

Abstract

The spatiotemporal (ST) surface has been shown to be a useful representation of projected scene dynamics. Our previous use of this representation has focused on geometric recovery of scene static structure from the analysis of relative motions on the moving image plane. That earlier work exploited the implicit partitioning of motions along epipolar lines to enable search-free feature tracking and position estimation. The ST manifolds provide explicit information about feature 3D contiguity, and their use leads to the recovery of feature 3D position, object 3D contours, and scene 3D surfaces. We have recently turned our attention to the task of interpreting non-static scenes, and track and estimate motions of independently moving objects and background by their appearance and behavior on the ST surface. Selecting the most reliable and discriminating information in the scene, the system demonstrates robust feature tracking over a large range of feature sizes and velocities. When coupled with the more mature Epipolar-Plane Image Analysis system, this motion analysis capability will enable camera solving, dynamics tracking, and scene reconstruction within a unified framework.

1 Image Sequence Tracking

The problem of tracking particular objects through a series of images has proved to be a challenging one. The most common tracking techniques include edge (or more generally, feature) tracking, centroid tracking, correlation tracking, and gradient-based optic flow analysis. Each suffers from significant disadvantages. Edge tracking is problematic because it is difficult to make a robust association between a particular group of edges[2] and the object being tracked. Centroid tracking is difficult for related reasons: there is no clear association between scene objects and computable centroids. Correlation tracking is problematic due to changing aspect of the target with respect to the tracker; the tracked object can rotate while translating, changing its image appearance from one frame to the next. Gradient-based optic flow relates differential changes in reflectance with orientation or motion of surfaces in the scene. This relationship is approximate for short spatial or temporal baselines and quite inappropriate for long baselines.[3] Feature analysis has advantage in that it focuses processing at the most discriminable parts of the imagery with the greatest localization and provides robustness through lower sensitivity to projective difficulties such as occlusion and illumination effects.

This paper describes our efforts at utilizing feature tracking on the space-time surface (2) for motion analysis. The principal distinction of this space-time-manifold approach to motion analysis is that it unifies the representation of scene features over space and time. In feature tracking this alleviates the major difficulty of feature-based analysis – the correspondence problem. For EPI analysis it will resolve independent motions within the same framework as solving for scene geometric structure. In discussing our approach to motion analysis we will begin by summarizing our earlier research in recovering scene structure from motion (Epipolar-Plane Image (EPI) Analysis, described in (3) and (4)), connect the techniques used there with the more general problem of unknown scene dynamics, and then discuss our use of the ST surface for motion tracking.

1.1 ST Manifolds for Scene Structure

In the scene reconstruction task, our use of the spatiotemporal surface involved tracking features as they moved under known constraints in space-time, and approximating and maintaining estimates of their positions through the sequence. The approach bridged the usual dichotomy of depth sensing in that its large number of images led to a large baseline and thus high accuracy, while rapid image sampling gave minimal change from frame to frame and, with camera knowledge, eliminated the correspondence problem. Within this framework, we generalized from the traditional notion of epipolar *lines*

[1] This paper appeared in the IEEE Motion Vision Workshop held in Princeton New Jersey in October 1991. The research described here has been supported by Fujitsu Systems Integration Laboratories most recently, with earlier support from DARPA.

[2] some of which may be only artifacts of position or illumination

[3] Notice the mapping and resampling necessary in Heel's work (9) to make optic flow coherent across time.

to that of epipolar *planes*. We then formulated a tracking process that exploited the above constraints in determining the position of features in the scene.

Our tracker was a sequential linear estimator, implemented as a Square Root filter without the extrapolation phase. Extrapolation was unnecessary since the camera constraints and the space-time surface told us where each feature moved from frame to frame (there was no 'aperture problem'). The work of Matthies(10) had similarities to ours in its pursuit of scene depth from the analysis of image sequences, but lacked several important elements, including the generality with respect to view angle that came with our use of a line-of-sight formulation, the explicit use of spatial connectivity that provides higher-level scene contour descriptors, and our match-free tracking.

While simplifying the problem through the use of three assumptions – the camera movement was linear, its position and attitude were known, data capture was sufficiently rapid that the imagery was temporally coherent – we developed a system that could a) work for any camera attitude, b) acquire images at varying rates, c) operate sequentially in time, and d) provide spatially coherent results — 3D contours.

Critical to this was the development of a unique process that constructed, in parallel as the frames were obtained, a 3D space-time description of the evolving imagery. This specifies fully the temporal and projective relationships between scene objects and the sensor. When the scene was stationary and observed by a moving camera, the representation provided simple, direct and robust estimates of scene structure. In extending the analysis of the ST surface representation for more general dynamic analysis, the major difference is that we cannot rely on known camera motion for our tracking, but must actually do the matching – addressing the correspondence problem. One of the benefits of the ST manifold is that it greatly simplifies this problem.

1.2 ST Manifolds for Scene Dynamics

Several issues arise in the move from static to dynamic scenes. Since we have to decouple sensor-induced motion from scene motion, we must be able to solve for the camera. For distinguishing moving from stationary objects, we must be able to discriminate real from sensor-induced motion (moving objects versus the background), – we must be able to model the scene static structure. Motion analysis and scene reconstruction should operate together, with the estimated scene geometry aiding in the camera solving (using known stationary features) and being used to discriminate object motion (by providing a 'background').

The approach we have taken to motion tracking is built on our scene structure estimation process within this unified framework. It's processing is based upon a multi-stage scheme involving feature detection, selection, grouping, and motion classification. First, we represent the spatiotemporal structure of scene dynamics – this is handled by the ST manifold. Using a localization measure on the space-time surface, we then isolate features of interest. Propagating from maxima of the localization measure, we determine the paths of features through time. Finally, a simple linear estimator characterizes feature velocity.

Feature 'edges' detected in 2D images become surface 'facets' in 3D. The connectivity of these facets gives us our tracking mechanism. We described earlier (2) how we locate and parameterize those individual 3D elements — the facets — and structure them together through time. In brief, we define the manifolds (sheets in space-time) that separate image features. These manifolds are 2D surfaces embedded in the 3D space-time dimensions of our data, and are positioned at the extrema of smoothed brightness gradient in the imagery – zero crossings of the Laplacian of a Gaussian (LOG). By following localizable 'features' on these surfaces we track them in time. The following sections describe our use of these features and provide details of our tracking method.

Although we have processed a variety of image sequences with this tracking process, display of detailed analyses of large data sets is difficult in panchromatic reproduction. Our displays here will be limited to simple local indications of the processing – more detail will be presented at the workshop, including various displays of the moving data; space-time surface building; representation of space-time surface localization; the trackers; the extracted 'interesting' features; superposition of reticles grouping observations on individual images in the sequences; and extraction of movement from the 'background.'

2 Selecting Tracking Features

Figure 1 left shows frames from a synthetic motion sequence of a rotating square. The motion is described by the zero-crossings of a 3D LOG over these data,[4] as shown in Figure 2, with time progressing out of the figure.[5] Figure 2 right shows a side view of these surfaces, oriented so that the temporal structure is more visible. What should be noticed is that the connectivity captured by the surface-building process is an explicit representation and grouping of the motion in the scene.

[4] 3D convolution is a standard means of incorporating temporal information (for example, see (1), (6) and (8)). In general, others have not attempted to utilitize or represent the temporal zero crossings.

[5] It is true; many of the figures presented here are so small and detailed as to seem unintelligible. Being of 3D data, larger single figures provide considerably less to appreciate and, in fact, viewing these in stereo gives very good assessment.

A spatial cut through the 3D zero crossings would produce a 2D spatial, single-image feature description. The temporal facets provide connectivity information for the space-time dimension of the data.

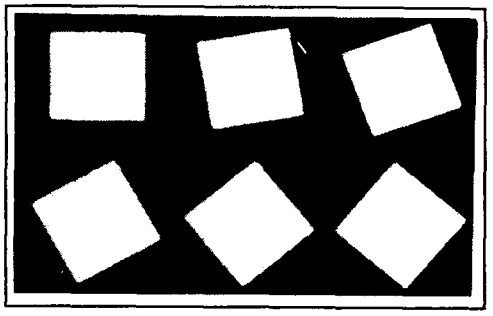


Fig. 1: Frames of Rotating Square

Tracking requires determining the correspondence between features in successive views. If we know the direction in which an object is moving (or conversely, the direction in which the sensor is moving through a static scene), then we can use this knowledge in determining their positions in space (as demonstrated by our EPI work). On the other hand, if we have no knowledge of the motion of the sensor, or if the scene can contain objects exhibiting independent motion, then these constraints do not apply. To track a feature we must be able to recognize it from frame to frame and distinguish it from the other features around it (this raises the aperture problem). Only a small percentage of the features we have selected with our 3D detection process can be adequately distinguished for this. For example, if the object to be tracked happens to have a square shape, then the only discriminable parts of it will be the corners. We must determine a measure to use on the images to locate features that *are* discriminable — features that can be reliably tracked from frame to frame.

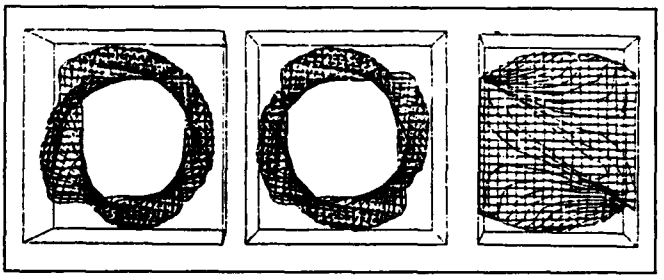


Fig. 2: Spatiotemporal Surfaces

2.1 The Autocorrelation Function

The autocorrelation function – convolving a small window of the image over some larger subset – provides such a measure. Where the window and the subset are identically aligned, the convolution will indicate a high correlation; elsewhere, the correlation will be poorer. A unimodal and highly peaked autocorrelation distribution indicates good localization, whereas a flat profile indicates ambiguity. Autocorrelation is quite expensive to compute, involving evaluation of order mn at every location in the image subset. Interpreting the autocorrelation structure is problematic.

2.2 Förstner's Measure

A variety of corner-detecting analogues to autocorrelation have been suggested, and we work with one developed by Förstner (7). Here, a simple measure based on a quotient of the determinant and the trace of the covariance matrix related to a planar fit to the window specifies the localizability of the feature at the center of the window. In its full development, the measure determines a confidence ellipse in which the feature can be expected to be localized. Three parameters of the measure define the major and minor axes of the ellipse and its orientation. These three parameters are mapped to a single value (FM).

Figure 3 shows a square and, beside it, the image of its FM. Figure 4 shows an amplified sampling of the localization measures for this image, and similar measures for the image rotated. These are oriented ellipses whose minor axis indicates the most reliable localization direction and whose axis magnitudes show the quality of the localization.

We have used two modifications to Förstner's measure in this work. We do not perform the center-of-gravity refinement, and we normalize the measure by the local gradient. The center-of-gravity computation improves the reliability of the estimation, especially in the vicinity of sharp corners where the simpler measure can produce dual peaks to the sides rather than a stronger peak at the vertex. Since we wish to develop a local computation mechanism for tracking, we are forgoing this correction in our initial studies.

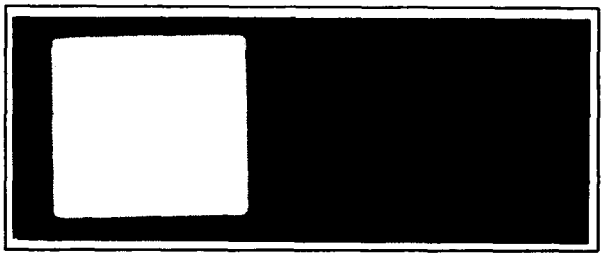


Fig. 3: Image Förstner Measure

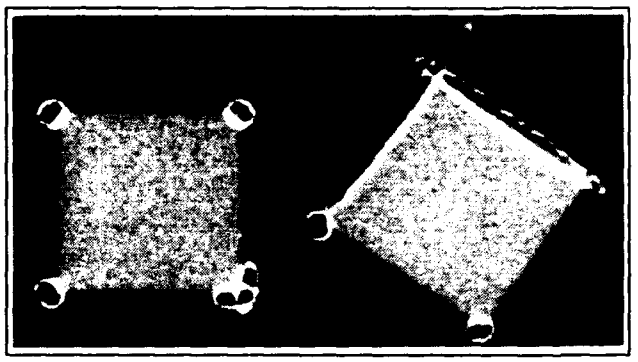


Fig. 4: Localization Ellipses

Figure 5 shows the structure of the space-time surface at the top right corner of the data, with dotted lines indicating the temporal connectivity. Figure 6 shows the relative strengths of the FM along this rotating corner, coded with dots and solid lines in increasing strength. It is clear that the corners of the square are the discriminable features, and the sides are poorly localized.

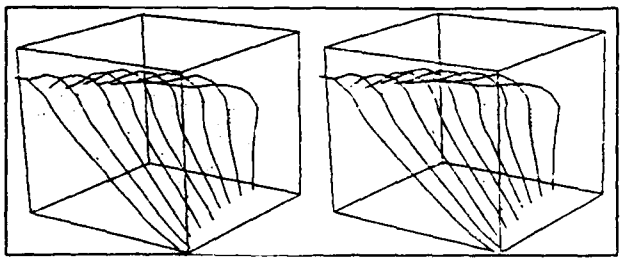


Fig. 5: Top Right Corner

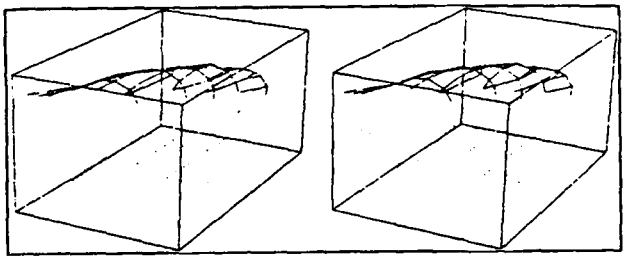


Fig. 6: FM Indications

2.3 Tracking and Velocity Estimation

The display in Figure 6 shows the features we will track — it does not show an actual tracking of features. Relating the observations indicated together over time must still be demonstrated. In forming these observations into unified trackers, we connect *local maxima* of these measures. The velocity estimation will then occur at this level of the analysis — as FM-maxima observations are associated through time, a sequential filter will be updated and refined with the new information.

2.3.1 Feature Tracking

The maxima can move in any direction between frames of a sequence, and can move an arbitrary distance depending on the velocity (translational and rotational) of the objects of which they are part. This means that while the spatiotemporal surface can be defined by feature connectivity, and a particular maximum will be seen to move along a single spatiotemporal surface, the matching of maxima cannot be defined strictly on the basis of proximity. For one thing, they need not be adjacent from frame to frame; for another, if maxima are fairly dense, then there may be significant difficulty in unambiguous assignment when they come close to one another. With large motions or repeated fine patterns, accurate tracking could be difficult.

As well as being accurate, our tracking mechanism must be designed to work within the framework of the surface-building process – it must be able to operate at a local level and be amenable to parallel implementation. The tracking mechanism we have designed satisfies these criteria. It uses a propagation mechanism, with each maximum at time T spreading itself forward to neighbors on the spatiotemporal surface and each maximum at time $T+1$ reaching back to neighbors on the spatiotemporal surface to see if there is a maximum from which it may have descended. When only one predecessor can be found, the tracking assignment can simply use this pairing, and can deduce that whatever the history of the feature at time T, this new observation at time $T+1$ shares that history and affects the estimation of that motion. When there are multiple choices for the assignment then an adjudication must be made to select the most likely. The tracking process and the adjudication are described in the next section.

2.3.2 Tracking Maxima on ST Surfaces

The principal intermediary in the tracking propagation is the set of *temporal* facets. When a contour is stationary, it's spatial facets will be adjacent over time – there will be no spatial motion requiring temporal representation. However, with spatial motion of a contour, the temporal facets serve to join these observations. At time $T+1$, the spatiotemporal surface from T to $T+1$ must be built, the local FM maxima be determined, and then be associated with previous FM maxima at T via the intervening temporal facets.

Several considerations are involved in establishing these temporal associations. If there is no ambiguity in the assignment (only one at each time is being considered for matching with the other), then the association is made and the tracking propagated. If more than one is in contention, then the values of the FM, the local spatial normal to the ST surface,[6] the established velocity, and the distance of motion are all considered. If one pairing is clearly better in position, orientation, and localization measure, then it is chosen, otherwise multiple

[6]Notice that the 3D surface normal also suggests the direction of feature motion, as would the eigenvector of the largest eigenvalue of a 3D version of Förstner's measure.

pairings are maintained as being possible. In the latter case, all contending tracks will be retained, with the final judgement being made later when more knowledge of the behavior of the features is available and a choice minimizing the ambiguity is possible. Another measure being evaluated is a correlation score (SSD) evaluated at contending ST facets. Figure 7 shows the tracking results for this rotating square.

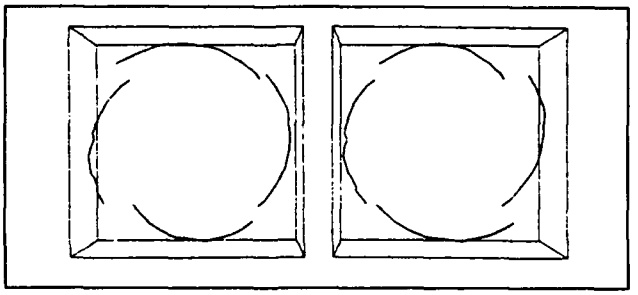


Figure 7: Trackers on Rotating Square

2.3.3 Motion Interpretation

Given a single feature moving in an image sequence, the best we can do, without other information (such as stereo or a DTM), is to determine its velocity in an image-based coordinate system. In our demonstrations here we will estimate only this image-plane velocity. As an initial approximation we will model feature motion as piecewise linear in time — that is, piecewise constant velocity.[7]

Image-plane motion is determined by solving a system of two linear equations defining the velocity vector in space-time. A sequential least-squares filter is set up for this estimation, and velocities are determined for each feature tracked. Velocity is a relative measure, and its interpretation depends on the activity around it. We must determine from the image-plane velocities which features are in motion with respect to others and which should be considered of importance for tracking – we must establish a background frame for velocity reference.

Since observed velocity depends on range, the geometric structure of the scene, if known, can be used to distinguish moving features from the background. Our earlier EPI work will provide the depth information for static components of the scene when it is integrated with this tracking system. In the meantime, our determination of the 'background' for these studies has been quite simple — we presume it to be planar and select as 'interesting' features those lying outside of one standard deviation from that estimated plane.

In the interests of demonstrating some preliminary object-like groupings, in the video demonstrations we isolate features moving with respect to the 'background' and group together those which are spatially connected. The reticles displayed over the imagery, indicating tracked features, enclose these grouped features. This demonstrates a primitive form of object tracking — features are tracked together when they are seen to be spatially related. Similar results were obtained by grouping features using their projective velocities. Our objective is to use behavior (dynamics) and shape (statics) to couple object tracking with object identification.

[7] See (5) for inferring a rotating and translating 3D model of object motion.

3 Experimental Results

Figure 8 shows several frames of part of a low resolution IR sequence of a moving car[8] – notice the vehicle moving down to the left while the background slides right because of a camera panning action. Figure 9 shows the major-contrast ST manifolds from these data. The next figure presents a side view, where the dots show temporal surface elements joining the solid lines of the spatial zero crossings. Figure 11 shows the final estimations, depicting velocities as dots through dashes to solid lines, in units of velocity standard deviation.

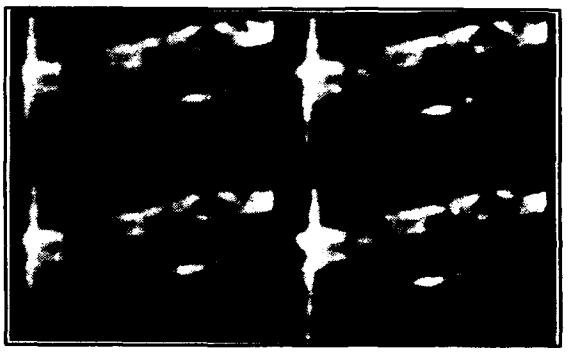


Figure 8: Moving Car

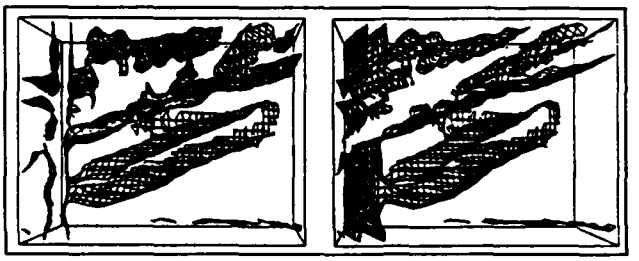


Figure 9: Spatiotemporal Surfaces

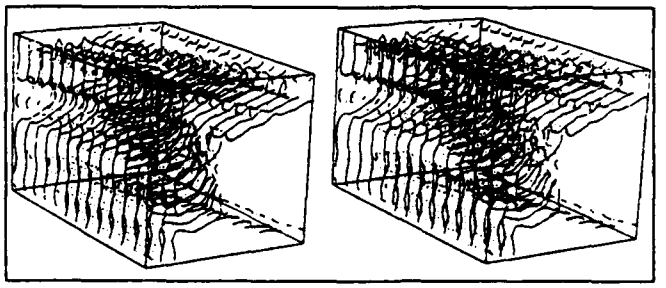


Figure 10: Side View of ST Surfaces

[8] These data are part of the workshop database.

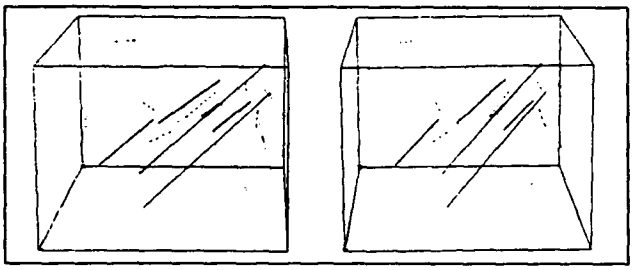


Figure 11: Velocity Estimates

3.1 Tracking Parameters

Recall that our first filtering of the imagery locates zero crossings of the Laplacian of a Gaussian (LOG) of the imagery. This operation selects the locally-largest intensity differences — the 'edges' in the images. The size of this filter determines the range of contrasts and the spatial extent of these edges. Although in this study we have selected large values for the filter size to facilitate display, these should be determined automatically by the tracking process – and determined differentially across the imagery depending on the character of the features observed. [9]

From the set of edges determined by the LOG operation, we select those whose gradients are greater than one standard deviation from the mean of gradients over all contours. This ensures that the features (edges) have 'significance,' i.e., are not likely to be artifacts of the detection process or noise. From among these gradient-selected features we choose those which are most localizable. In selecting from among the most localizable features that are tracked for those we wish to consider of 'interest,' we have again used some *a priori* assumptions. Notable among these is the assumption that such features are moving with respect to the background (our EPI analysis will handle the converse). We also require, for reliable tracking, that we have enough observations of a feature to enable an accurate and consistent estimate of its velocity to be made. This means that we discard (for our displays) tracked features that are not viewed for a sufficient duration.

3.2 Performance Issues

The current surface-building process constructs the spatiotemporal representations of the imagery at a rate of about 1000 pixels per second. Evaluation of Förstner's measure and the tracking of FM maxima reduce this to roughly 500 pixels per second. Gaussian and Laplacian convolution are not included in these figures since we compute the filtered images in an off-line fashion before studying a data set. These convolution computations are quite simple, however, as they are decomposable into a total of eleven 1D convolutions, and could be computed in a realtime pipeline. The tracking has been designed with parallel-processing in mind, and most computations require only a small local support. Such parallelism could provide sufficient performance for real-time analysis.

[9] We are addressing the notion of scale filtering on the scale-space manifold, and are finding that analysis over a range of resolutions can lead to selecting the 'best' filter size for each feature.

3.3 Concluding Remarks

Our intented use of this tracking process begins with estimation of the dynamics of objects in motion and their subsequent recognition based on behavioral and shape characteristics. We will also be integrating the tracker with the original EPI analysis. Features determined to be stationary will be used for camera solving, and this will enable processing of non-linear camera motions. In an intriguing combination of the two, we are investigating use of derived groupings and rigid motion interpretations to run the EPI analysis in reverse over the space-time surfaces (using inverses of the observed motion parameters), and compute the 3D shape of tracked objects even while they are undergoing independent and arbitrary motion.

The important element to note in this work is not that we can track features through a sequence – there are a variety of techniques that can do this, more or less successfully – but that we can utilize the ST surfaces to track and estimate feature motions, distinquish moving from stationary objects, and inform EPI analysis of features to use for camera solving in its geometric recovery. This is a critical component of enabling EPI analysis to be used on non-linear motion trajectories through dynamic scenes.

References

[1] Adelson, E.H., and J.R. Bergen, "Spatiotemporal energy models for the perception of motion," *Journal of the Optical Society of America A*, **2**(2): 284–299, 1985 .

[2] Baker, H.H., "Building Surfaces of Evolution: The Weaving Wall," *International Journal of Computer Vision*, **3**(1): 51–72, 1989.

[3] Baker, H.H., and R.C. Bolles, "Generalizing Epipolar-Plane Image Analysis on the Spatiotemporal Surface," *International Journal of Computer Vision*, **3**(1): 33–50, 1989.

[4] Bolles, R.C., H.H. Baker, and D.H. Marimont, "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion," *International Journal of Computer Vision*, **1**(1): 7–55, 1987.

[5] Broida, T.J., and R. Chellappa, "Estimating the Kinematics and Structure of a Rigid Object from

a Sequence of Monocular Images," *IEEE PAMI*, 13(6): 497–513, 1991.

[6] Buxton, B.F., and H. Buxton, "Computation of Optic Flow from the Motion of Edge Features in Image Sequences," *Image and Vision Computing*, 2(2): 59–75, 1984.

[7] Förstner, W., "Reliability Analysis of Parameter Estimation in Linear Models with Applications to Mensuration Problems in Computer Vision," *Computer Vision, Graphics, and Image Processing*, 40: 273–310, 1987.

[8] Heeger, D.J., "Depth and Flow from Motion Energy," *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, 657–663, 1986.

[9] Heel, J., "Temporally Integrated Surface Reconstruction," *Proceedings of the Third International Conference on Computer Visionn*, IEEE Computer Society, Osaka, Japan, 292–295, 1990.

[10] Matthies, L., R. Szeliski, and T. Kanade, "Incremental Estimation of Dense Depth Maps from Image Sequences," *Proceedings of the Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, Ann Arbor, Michigan, 366–374, 1988.

The Factorization Method for the Recovery of Shape and Motion from Image Streams

Carlo Tomasi
Department of Computer Science
Cornell University
Ithaca, NY 14850

Takeo Kanade
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Recovering scene geometry and camera motion from a stream of images is an important problem in computer vision. If the scene geometry is specified by depth measurements, that is, by camera-to-scene distances, noise sensitivity worsens rapidly with increasing depth.

In this paper, we show that this difficulty can be overcome by computing scene geometry directly in terms of shape, that is, by computing the coordinates of feature points in the scene with respect to a world-centered system, without recovering camera-centered depth as an intermediate quantity. More specifically, we show that a matrix of image measurements can be factored by Singular Value Decomposition into the product of two matrices that represent shape and motion, respectively.

After describing this *factorization method*, we extend it to deal with feature occlusions. We demonstrate the accuracy and robustness of the method with a series of experiments on laboratory as well as outdoor streams, with and without occlusions.

1 Introduction

Recovering scene geometry and camera motion from a stream of images is an important problem in computer vision. It admits a solution [Ull79], [RA79] for perfect images, but is very sensitive to noise [BJT90]. In this paper, we observe that this sensitivity is due in part to the representation of shape as a depth map, and show that reformulating the problem in world-centered coordinates leads to a simpler and better-behaved solution.

In Ullman's proof of existence of a solution [Ull79], as well as in the perspective formulation in [RA79], the coordinates of feature points in the world are expressed in a world-centered coordinate system.

However, this choice has been replaced by most computer vision researchers with that of a camera-centered representation of shape [Pra80], [BH83], [TH84], [Adi85], [WW85], [BBM87], [HHN88], [HJ89], [Hee89], [MKS89], [SA89], [BCC90]. With this representation, the position of feature points is specified by their image coordinates and by their depths, defined as the distances between the camera center and the feature points, measured along the optical axis.

Unfortunately, although a camera-centered representation simplifies the equations for perspective projection, it makes shape estimation harder and, for increasingly distant scenes, eventually impossible. This is due to two reasons. First, the computation of shape via depth is very sensitive to noise for remote objects: since even large changes in depth produce small changes in the image, computing small depth differences from image variations is virtually impossible with any amount of image noise.

Second, as the camera moves, the camera-centered feature coordinates change. This leads to the difficult problem of relating depth values in different camera coordinate systems to each other in the presence of motion uncertainties (see for instance [MKS89], [Hee89]).

In this paper, we show that both difficulties disappear if feature coordinates are expressed with respect to a world-centered frame. With this formulation, object-centered shape can be linked to image motion directly, without using depth as an intermediate quantity. Furthermore, the mutual independence of shape and motion in world-centered coordinates makes it possible to cast structure-from-motion as the problem of factoring a matrix of image measurements into the product of two matrices that represent respectively the camera rotation and the shape of the scene.

More specifically, an image stream can be represented by a $2F \times P$ measurement matrix which gathers the horizontal and vertical coordinates of P points tracked through F frames. If image coordinates are measured with respect to their centroid, we prove the following *rank theorem*: under orthography, the measurement matrix is of rank 3. As a consequence of this theorem, we show that the measurement matrix can be factored into the product of two matrices of size $2F \times 3$ and $3 \times P$, respectively, where the first matrix encodes camera rotation, the second shape.

The rank theorem captures precisely the nature of the redundancy of an image stream, and allows dealing with a large number of points and frames in a conceptually simple and computationally efficient way to reduce the effects of noise. The resulting algorithm is based on the Singular Value Decomposition, which is numerically well-behaved and stable.

We first introduced this factorization method in [TK90], where we treated the simple case of single-scanline images in a flat, two-dimensional world. We now develop the idea into a working system for arbitrary camera motion in three dimensions and full two-dimensional images.

An approach related to ours, but using a different formalism, appeared in [DA90]. Debrunner and Ahuja supply both closed-form expressions for shape and motion and an incremental solution (one image at a time). The price they pay for these advantages is to assume that motion is constant. It will be interesting to compare how easily our approach and theirs can be generalized.

In the next section we show how to build the measurement matrix from an image stream, prove that the measurement matrix is of rank 3, and show how to use this result to factor the measurement matrix into shape and camera rotation. Section 3 describes an experiment on a real image stream.

This stream was produced under carefully controlled laboratory conditions. However, lab images are too clean to be fully realistic. Furthermore, only features that are visible in all the frames are used in the stream of section 5. In reality, as the camera moves, features can appear and disappear from the image, because of occlusions. This phenomenon is frequent enough to make a shape and motion computation method unrealistic if it cannot deal with it.

In sections 4 and 5 we address these concerns by extending the factorization method to streams with occlusions and by testing the method with outdoor streams recorded on videotape with an amateur camera. Specifically, in section 4 we introduce the extension to occlusions and we test it with another lab experiment. Finally, in section 5 we show the results for two outdoor videotape streams, one in which occlusions are negligible, the other in which they are dominant.

2 The Factorization Method

In the next subsection we show how to represent an image stream as a matrix of image feature coordinates. We then introduce the main result on the rank of the measurement matrix in the absence (subsection 2.2) and presence (subsection 2.3) of noise. Subsection 2.4 shows that the motion and shape result is essentially unique, and subsection 2.5 summarizes the factorization method.

To track features from frame to frame, we used a method based on [LK81] which we extended to allow for the automatic selection of features. The description of both detection and tracking are beyond the scope of this paper.

2.1 The Measurement Matrix

If we track P feature points over F frames in the image stream, we obtain a stream of image coordinates $\{(u_{fp}, v_{fp}) \mid f = 1, \ldots, F, p = 1, \ldots, P\}$.

The horizontal feature coordinates u_{fp} are written into an $F \times P$ matrix U: there is one row per frame, and one column per feature point. Similarly, an $F \times P$ matrix V is built from the vertical coordinates v_{fp}.

The rows of the matrices U and V are then registered by subtracting from each entry the mean of the entries in the same row:

$$\begin{aligned} \tilde{u}_{fp} &= u_{fp} - a_f \\ \tilde{v}_{fp} &= v_{fp} - b_f \,, \end{aligned} \tag{1}$$

where

$$\begin{aligned} a_f &= \frac{1}{P}\sum_{p=1}^{P} u_{fp} \\ b_f &= \frac{1}{P}\sum_{p=1}^{P} v_{fp} \,. \end{aligned}$$

This produces two new $F \times P$ matrices $\tilde{U} = [\tilde{u}_{fp}]$ and $\tilde{V} = [\tilde{v}_{fp}]$. The matrix

$$\widetilde{W} = \left[\frac{\tilde{U}}{\tilde{V}}\right]$$

is called the *registered measurement matrix*. This is the input to our shape-and-motion algorithm.

2.2 The Rank Theorem

We now analyze the relation between camera motion, shape, and the entries of the registered measurement matrix $\widetilde{W}$. This analysis leads to the key result that $\widetilde{W}$ is highly rank-deficient (the rank theorem).

The orientation of the camera reference system corresponding to frame number f is determined by a pair of unit vectors, $\mathbf{i}_f$ and $\mathbf{j}_f$, pointing along the scanlines and the columns of the image respectively, and defined with respect to a world reference system with coordinates x, y, and z (see Figure 1). Under orthography, all projection rays are then parallel to the cross product of $\mathbf{i}_f$ and $\mathbf{j}_f$:

$$\mathbf{k}_f = \mathbf{i}_f \times \mathbf{j}_f \,.$$

The origin of the camera reference system is at the center of the image, while the origin of the world is the centroid of the points $\mathbf{s}_p = (x_p, y_p, z_p)^T$ in space, so that

$$\frac{1}{P}\sum_{p=1}^{P} \mathbf{s}_p = 0 \,.$$

From Figure 1 we see that the projection (u_{fp}, v_{fp}) of point $\mathbf{s}_p = (x_p, y_p, z_p)^T$ onto frame f is given by the equations

$$\begin{aligned} u_{fp} &= \mathbf{i}_f{}^T(\mathbf{s}_p - \mathbf{t}_f) \\ v_{fp} &= \mathbf{j}_f{}^T(\mathbf{s}_p - \mathbf{t}_f) \,, \end{aligned}$$

where $\mathbf{t}_f = (a_f, b_f, c_f)^T$ is the vector from the world origin to the image center of frame f.

We can now write expressions for the entries $\tilde{u}_{fp}$ and $\tilde{v}_{fp}$ of the registered measurement matrix by substituting the projection equations above into the registration equations (1). For the horizontal coordinates we have

$$\begin{aligned} \tilde{u}_{fp} &= u_{fp} - a_f \\ &= \mathbf{i}_f{}^T(\mathbf{s}_p - \mathbf{t}_f) - \frac{1}{P}\sum_{q=1}^{P} \mathbf{i}_f{}^T(\mathbf{s}_q - \mathbf{t}_f) \end{aligned}$$

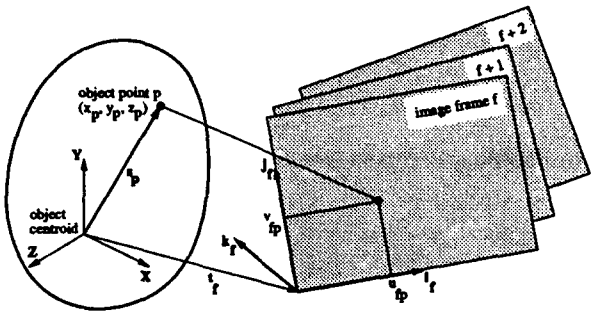


Figure 1: The systems of reference used in our problem formulation.

$$= \mathbf{i}_f^T \left(\mathbf{s}_p - \frac{1}{P} \sum_{q=1}^{P} \mathbf{s}_q \right)$$

$$= \mathbf{i}_f^T \mathbf{s}_p \ . \qquad (2)$$

We can write a similar equation for the registered vertical image projection $\widetilde{v}_{fp}$. To summarize,

$$\begin{aligned} \widetilde{u}_{fp} &= \mathbf{i}_f^T \mathbf{s}_p \\ \widetilde{v}_{fp} &= \mathbf{j}_f^T \mathbf{s}_p \ . \end{aligned} \qquad (3)$$

Because of the two sets of $F \times P$ equations (3), the registered measurement matrix $\widetilde{W}$ can be expressed in a matrix form:

$$\widetilde{W} = RS \qquad (4)$$

where

$$R = \begin{bmatrix} \mathbf{i}_1^T \\ \vdots \\ \mathbf{i}_F^T \\ \mathbf{j}_1^T \\ \vdots \\ \mathbf{j}_F^T \end{bmatrix} \qquad (5)$$

represents the camera rotation, and

$$S = [\ \mathbf{s}_1 \ \cdots \ \mathbf{s}_P\ ] \qquad (6)$$

is the shape matrix. In fact, the rows of R represent the orientations of the horizontal and vertical camera reference axes throughout the stream, while the columns of S are the coordinates of the P feature points with respect to their centroid.

From the first and the last line of equation (2), the original unregistered matrix W can be written as

$$W = RS + \mathbf{t}\mathbf{e}_P^T \ , \qquad (7)$$

where $\mathbf{t} = (a_1, \ldots, a_F, b_1, \ldots, b_F)^T$ is a $2F$-dimansional vector that gathers the projections of camera translation along the image plane (see equation (2). In scalar form,

$$\begin{aligned} u_{fp} &= \mathbf{i}_f^T \mathbf{s}_p + a_f \\ v_{fp} &= \mathbf{j}_f^T \mathbf{s}_p + b_f \ . \end{aligned} \qquad (8)$$

In the equations above, $\mathbf{i}_f$ and $\mathbf{j}_f$ are mutually orthogonal unit vectors, so they must satisfy the constraints

$$|\mathbf{i}_f| = |\mathbf{j}_f| = 1 \quad \text{and} \quad \mathbf{i}_f^T \mathbf{j}_f = 0 \ . \qquad (9)$$

Also, the rotation matrix R is unique if the system of reference for the solution is aligned, say, with that of the first camera position:

$$\mathbf{i}_1 = (1,0,0)^T \quad \text{and} \quad \mathbf{j}_1 = (0,1,0)^T \ . \qquad (10)$$

Since R is $2F \times 3$ and S is $3 \times P$, the matrix projection equation (4) implies the following **rank theorem**.

> *Without noise, the registered measurement matrix $\widetilde{W}$ is at most of rank three.*

The rank theorem expresses the fact that the $2F \times P$ image measurements are highly redundant. Indeed, they could all be described concisely by giving F frame reference systems and P point coordinate vectors, if only these were known.

When noise corrupts the images, the registered measurement matrix $\widetilde{W}$ will not be exactly of rank 3. However, the rank theorem can be extended to the case of noisy measurements in a well-defined manner. The next subsection introduces this extension, using the concept of Singular Value Decomposition [GR71] to introduce the notion of approximate rank.

2.3 Approximate Rank

Assuming [1] that $2F \geq P$, the matrix $\widetilde{W}$ can be decomposed [GR71] into a $2F \times P$ matrix O_1, a diagonal $P \times P$ matrix Σ, and a $P \times P$ matrix O_2,

$$\widetilde{W} = O_1 \Sigma O_2 \ , \qquad (11)$$

such that $O_1^T O_1 = O_2^T O_2 = O_2 O_2^T = \mathcal{I}$, and $\sigma_1 \geq \ldots \geq \sigma_P$. Here, $\mathcal{I}$ is the $P \times P$ identity matrix, and the *singular values* $\sigma_1, \ldots, \sigma_P$ are the diagonal entries of Σ. This is the *Singular Value Decomposition* (SVD) of the matrix $\widetilde{W}$.

If we now partition the matrices O_1, Σ, and O_2 as follows:

$$\begin{aligned} O_1 &= \underbrace{[\ O_1' \ |}_{3} \ \underbrace{O_1''\ ]}_{P-3} \}2F \\ \Sigma &= \underbrace{\left[\begin{array}{c|c} \Sigma' & 0 \\ \hline 0 & \Sigma'' \end{array}\right]}_{\begin{array}{cc} 3 & P-3 \end{array}} \begin{array}{l} \}3 \\ \}P-3 \end{array} \\ O_2 &= \underbrace{\left[\begin{array}{c} O_2' \\ \hline O_2'' \end{array}\right]}_{P} \begin{array}{l} \}3 \\ \}P-3 \end{array} \ , \end{aligned} \qquad (12)$$

we have

$$O_1 \Sigma O_2 = O_1' \Sigma' O_2' + O_1'' \Sigma'' O_2'' \ .$$

Let $\widetilde{W}^*$ be the ideal registered measurement matrix, that is, the matrix we would obtain in the absence of noise. Because of the rank theorem, $\widetilde{W}^*$ has at most

[1] This assumption is not crucial: if $2F < P$, everything can be repeated for the transpose of $\widetilde{W}$.

three non-zero singular values. Since the singular values in Σ are sorted in non-increasing order, Σ' must contain all the singular values of $\widetilde{W}^*$ that exceed the noise level. As a consequence, the term $O_1''\Sigma''O_2''$ must be due entirely to noise, and the product $O_1'\Sigma'O_2'$ is the best possible rank-3 approximation to $\widetilde{W}^*$.

We can now restate our **rank theorem for noisy measurements**.

> *All the shape and rotation information in $\widetilde{W}$ is contained in its three greatest singular values, together with the corresponding left and right eigenvectors.*

Thus, the best possible approximation to the ideal registered measurement matrix $\widetilde{W}^*$ is the product

$$\hat{W} = O_1'\Sigma'O_2'$$

where the primes refer to the partition (12). With the definitions

$$\begin{aligned} \hat{R} &= O_1'[\Sigma']^{1/2} \\ \hat{S} &= [\Sigma']^{1/2}O_2' , \end{aligned}$$

we can also write

$$\hat{W} = \hat{R}\hat{S} . \tag{13}$$

The two matrices $\hat{R}$ and $\hat{S}$ are of the same size as the desired rotation and shape matrices R and S: $\hat{R}$ is $2F \times 3$, and $\hat{S}$ is $3 \times P$. However, the decomposition (13) is not unique. In fact, if Q is *any* invertible 3×3 matrix, the matrices $\hat{R}Q$ and $Q^{-1}\hat{S}$ are also a valid decomposition of $\hat{W}$, since

$$(\hat{R}Q)(Q^{-1}\hat{S}) = \hat{R}(QQ^{-1})\hat{S} = \hat{R}\hat{S} = \hat{W} .$$

Thus, $\hat{R}$ and $\hat{S}$ are in general different from R and S. A striking fact, however, is that except for noise the matrix $\hat{R}$ is a linear transformation of the true rotation matrix R, and the matrix $\hat{S}$ is a linear transformation of the true shape matrix S. Indeed, in the absence of noise, R and $\hat{R}$ both span the column space of the registered measurement matrix $\widetilde{W} = \widetilde{W}^* = \hat{W}$. Since that column space is three-dimensional because of the rank theorem, R and $\hat{R}$ are different bases for the same space, and there must be a linear transformation between them.

Whether the noise level is low enough that it can be ignored at this juncture depends also on the camera motion and on shape. Notice, however, that the singular value decomposition yields sufficient information to make this decision: the requirement is that the ratio between the third and the fourth largest singular values of $\widetilde{W}$ be sufficiently large.

2.4 The Metric Constraints

To summarize, the matrix $\hat{R}$ is a linear transformation of the true rotation matrix R. Likewise, $\hat{S}$ is a linear transformation of the true shape matrix S. More specifically, there exists a 3×3 matrix Q such that

$$\begin{aligned} R &= \hat{R}Q \\ S &= Q^{-1}\hat{S} . \end{aligned} \tag{14}$$

In order to find Q we observe that the rows of the true rotation matrix R are unit vectors and the first F are orthogonal to corresponding F in the second half of R. These *metric constraints* yield the over-constrained, quadratic system

$$\begin{aligned} \hat{\mathbf{i}}_f^T QQ^T\hat{\mathbf{i}}_f &= 1 \\ \hat{\mathbf{j}}_f^T QQ^T\hat{\mathbf{j}}_f &= 1 \\ \hat{\mathbf{i}}_f^T QQ^T\hat{\mathbf{j}}_f &= 0 \end{aligned} \tag{15}$$

in the entries of Q. This is a simple data fitting problem which, though nonlinear, can be solved efficiently and reliably.

A last ambiguity needs to be resolved: if Q is a solution of the metric constraint problem, so is QN, where N is any orthonormal matrix. In fact,

$$\begin{aligned} \hat{\mathbf{i}}_f^T(QN)(N^TQ^T)\hat{\mathbf{i}}_f &= \hat{\mathbf{i}}_f^T Q(NN^T)Q^T\hat{\mathbf{i}}_f \\ &= \hat{\mathbf{i}}_f^T QQ^T\hat{\mathbf{i}}_f \\ &= 1 , \end{aligned}$$

and likewise for the remaining two constraint equations. Geometrically, this corresponds to the fact that the solution is determined up to a rotation, since the orientation of, say, the first camera reference system with respect to the world reference system is arbitrary. This arbitrariness can be removed, if desired, by rotating the solution so that the first frame is represented by the identity matrix.

2.5 Outline of the Complete Algorithm

Based on the development in the previous sections, we now have a complete algorithm for the computation of shape and rotation from the registered measurement matrix $\widetilde{W}$ derived from a stream of images. To summarize, the rotation matrix R and the shape matrix S defined in equations (5) and (6) can be computed as follows.

1. Compute the singular-value decomposition $\widetilde{W} = O_1\Sigma O_2$.
2. Define $\hat{R} = O_1'(\Sigma')^{1/2}$ and $\hat{S} = (\Sigma')^{1/2}O_2'$, where the primes refer to the block partitioning defined in (12).
3. Compute the matrix Q in equations (14) by imposing the metric constraints (equations (15)).
4. Compute the rotation matrix R and the shape matrix S as $R = \hat{R}Q$ and $S = Q^{-1}\hat{S}$.
5. If desired, align the first camera reference system with the world reference system by forming the products RR_0 and R_0^TS, where the orthonormal matrix $R_0 = [\mathbf{i}_1\ \mathbf{j}_1\ \mathbf{k}_1]$ rotates the first camera reference system into the identity matrix.

3 An Experiment

In this section we test the factorization method with an experiment on a real stream of images. The images depict a small plastic model of a building. The camera is a Sony CCD camera with a 200 mm lens, and is moved

by means of a high-precision positioning platform. Some frames in the stream are shown in figure 3. Camera pitch, yaw, and roll around the model are all varied as shown by the dashed curves in figure 4. The translation of the camera is such as to keep the building within the field of view of the camera.

For feature tracking, we extended the method described in [LK81] to allow also for the automatic selection of image features. The entire set of 430 features is displayed in figure 5, overlaid on the first frame of the stream. Of these features, 42 were abandoned during tracking because their appearance changed too much. The remaining 388 features are used in the computation of shape and motion.

The plots in figure 4 compare the rotation components computed by the algorithm (solid curves) with the values measured mechanically from the mobile platform (dashed curves). The differences are magnified in figure 6. The errors are everywhere less than 0.4 degrees. The computed motion follows closely also rotations with curved profiles, such as the roll profile between frames 1 and 20 (second plot in figure 4), and faithfully preserves all discontinuities in the rotational velocities: the algorithm does not smooth the results.

Between frames 60 and 80, yaw and pitch are nearly constant, that is, the camera merely rotates about its optical axis. This demonstrates that it is sufficient for the stream *as a whole* to be taken during nondegenerate motion. The algorithm can deal without difficulty with streams that contain degenerate substreams, because the information in the stream is used all at once in our method.

The shape results are shown qualitatively in figure 7, which shows the computed shape viewed from above. The view in figure 7 is similar to that in figure 8, included for visual comparison. Notice that the walls, the windows on the roof, and the chimneys are recovered in their correct positions.

To evaluate the shape performance quantitatively, we measured some distances on the actual house model with a ruler, and compared them with the distances computed from the point coordinates in the shape results. Figure 9 shows the selected features. The diagram in figure 10 shows the distances between pairs of features measured on the actual model and computed by our algorithm. The measured distances between the steps along the right side of the roof (7.2 mm) were obtained by measuring five steps and dividing the total distance (36 mm) by five. The differences between computed and measured results are of the order of the resolution of our ruler measurements (one millimeter).

4 Occlusions

Two separate problems arise when features are occluded during the image stream: how to detect occlusions and how to deal with them once they occur. A method to detect occlusions was suggested in [TK91b], based on monitoring the photometric difference between a given feature in the first and in the current frame. Work on that method is still in progress. In this section, we deal with the second problem above and assume that occlusion events are given. In the experiments below, occlusions are marked by hand.

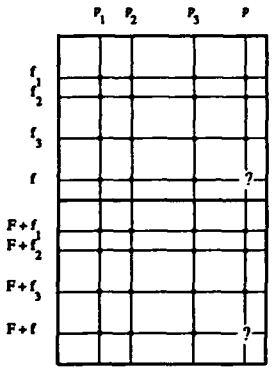


Figure 2: The Reconstruction Condition. If the dotted entries of the measurement matrix are known, the two unknown ones (question marks) can be reconstructed.

Sequences with appearing and disappearing features result in a partially unknown measurement matrix W, so that the factorization method introduced in section 2 cannot be applied directly.

However, there is usually sufficient information in the stream to determine all the camera positions and all the three-dimensional feature point coordinates. If that is the case, we can not only solve the shape and motion recovery problem from the incomplete measurement matrix W, but we can even hallucinate the unknown entries of W by projecting the computed three-dimensional feature coordinates onto the computed camera positions.

In this section, we first introduce a solution *propagation method* to grow a shape and motion solution for a full submatrix of W into a solution for all of W (subsection 4.1). We extend the propagation method to noisy streams in subsection 4.2. In section 5.1 we demonstrate the method on a real stream of images, in which a ping-pong ball is rotated more than 360 degrees in front of the camera.

4.1 Solution for Noise-Free Images

In this subsection, we show how to fill in the unknown entries of the measurement matrix in the case of noise-free images. In the next subsection, we extend the method to noisy streams.

If there is no noise, the occlusion problem is solvable if every unknown image feature coordinate belongs to an otherwise full 4×4 submatrix of W. Formally, we have the following sufficient condition.

Condition for Reconstruction In the absence of noise, an unknown image measurement pair (u_{fp}, v_{fp}) can be reconstructed if point p is visible in at least three more frames f_1, f_2, f_3, and if there are at least three more points p_1, p_2, p_3 that are visible in frames f, f_1, f_2, f_3.

Referring to Figure 2, this means that the dotted entries must be known to reconstruct the question marks. In this subsection, we prove this condition. To this end, we notice that the rows and columns of the noise-free

measurement matrix W can always be permuted so that $f_1 = p_1 = 1$, $f_2 = p_2 = 2$, $f_3 = p_3 = 3$, $f = p = 4$. We can therefore suppose that u_{44} and v_{44} are the only two unknown entries in the 8×4 matrix

$$W = \begin{bmatrix} U \\ \hline V \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ u_{21} & u_{22} & u_{23} & u_{24} \\ u_{31} & u_{32} & u_{33} & u_{34} \\ u_{41} & u_{42} & u_{43} & ? \\ v_{11} & v_{12} & v_{13} & v_{14} \\ v_{21} & v_{22} & v_{23} & v_{24} \\ v_{31} & v_{32} & v_{33} & v_{34} \\ v_{41} & v_{42} & v_{43} & ? \end{bmatrix} .$$

Then, the factorization method can be applied to the first three rows of U and V, that is, to the 6×4 submatrix

$$W_{6\times4} = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ u_{21} & u_{22} & u_{23} & u_{24} \\ u_{31} & u_{32} & u_{33} & u_{34} \\ v_{11} & v_{12} & v_{13} & v_{14} \\ v_{21} & v_{22} & v_{23} & v_{24} \\ v_{31} & v_{32} & v_{33} & v_{34} \end{bmatrix} \qquad (16)$$

to produce the partial translation and rotation submatrices

$$\mathbf{t}_{6\times1} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad \text{and} \quad R_{6\times3} = \begin{bmatrix} \mathbf{i}_1^T \\ \mathbf{i}_2^T \\ \mathbf{i}_3^T \\ \mathbf{j}_1^T \\ \mathbf{j}_2^T \\ \mathbf{j}_3^T \end{bmatrix} \qquad (17)$$

and the full shape matrix

$$S = [\ \mathbf{s}_1 \quad \mathbf{s}_2 \quad \mathbf{s}_3 \quad \mathbf{s}_4\ ] \qquad (18)$$

such that

$$W_{6\times4} = R_{6\times3}S + \mathbf{t}_{6\times1}\mathbf{e}_4^T$$

($\mathbf{e}_4^T$ is a row vector of four ones).

To complete the rotation solution, we need to compute the vectors $\mathbf{i}_4$ and $\mathbf{j}_4$. However, a registration problem must be solved first. In fact, only three points are visible in the fourth frame, while equation (18) yields all four points in space. Since the factorization method computes the space coordinates with respect to the centroid of the points, we have $\mathbf{s}_1 + \mathbf{s}_2 + \mathbf{s}_3 + \mathbf{s}_4 = 0$, while the image coordinates in the fourth frame are measured with respect to the centroid of just three points (1, 2, 3). Thus, before we can compute $\mathbf{i}_4$ and $\mathbf{j}_4$ we must make the two origins coincide by referring all coordinates to the centroid

$$\mathbf{c} = \frac{1}{3}(\mathbf{s}_1 + \mathbf{s}_2 + \mathbf{s}_3)$$

of the three points that are visible in all four frames. In the fourth frame, the projection of c has coordinates

$$\begin{aligned} a'_4 &= \frac{1}{3}(u_{41} + u_{42} + u_{43}) \\ b'_4 &= \frac{1}{3}(v_{41} + v_{42} + v_{43}) , \end{aligned}$$

so we can define the new coordinates

$$\mathbf{s}'_p = \mathbf{s}_p - \mathbf{c} \quad \text{for} \quad p = 1,2,3$$

in space and

$$\begin{aligned} u'_{4p} &= u_{4p} - a'_4 \\ v'_{4p} &= v_{4p} - b'_4 \end{aligned} \quad \text{for} \quad p = 1,2,3$$

in the fourth frame. Then, $\mathbf{i}_4$ and $\mathbf{j}_4$ are the solutions of the two 3×3 systems

$$\begin{aligned} [\ u'_{41} \quad u'_{42} \quad u'_{43}\ ] &= \mathbf{i}_4^T [\ \mathbf{s}'_1 \quad \mathbf{s}'_2 \quad \mathbf{s}'_3\ ] \\ [\ v'_{41} \quad v'_{42} \quad v'_{43}\ ] &= \mathbf{j}_4^T [\ \mathbf{s}'_1 \quad \mathbf{s}'_2 \quad \mathbf{s}'_3\ ] \end{aligned} \qquad (19)$$

derived from equation (4). The second equation in (17) and the solution to (19) yield the entire rotation matrix R, while shape is given by equation (18).

The components a_4 and b_4 of translation in the fourth frame with respect to the centroid of all four points can be computed by postmultiplying equation (7) by the vector $\eta_4 = (1,1,1,0)^T$:

$$W\eta_4 = RS\eta_4 + \mathbf{t}\mathbf{e}_4^T\eta_4 .$$

Since $\mathbf{e}_4^T\eta_4 = 3$, we obtain

$$\mathbf{t} = \frac{1}{3}(W - RS)\eta_4 . \qquad (20)$$

In particular, rows 4 and 8 of this equation yield a_4 and b_4. Notice that the unknown entries u_{44} and v_{44} are multiplied by zeros in equation (20).

Now that both motion and shape are known, the missing entries u_{44}, v_{44} of the measurement matrix W can be found by orthographic projection (equation (8)):

$$\begin{aligned} u_{44} &= \mathbf{i}_4^T\mathbf{s}_4 + a_4 \\ v_{44} &= \mathbf{j}_4^T\mathbf{s}_4 + b_4 . \end{aligned}$$

The procedure thus completed factors the full 6×4 submatrix of W and then reasons on the three points that are visible in all the frames to compute motion for the fourth frame. Alternatively, one can first apply factorization to the 8×3 submatrix

$$W_{8\times3} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \\ u_{41} & u_{42} & u_{43} \\ v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \\ v_{41} & v_{42} & v_{43} \end{bmatrix} \qquad (21)$$

to produce the full translation and rotation submatrices

$$\mathbf{t}' = \begin{bmatrix} a'_1 \\ a'_2 \\ a'_3 \\ a'_4 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{bmatrix} \quad \text{and} \quad R = \begin{bmatrix} \mathbf{i}_1^T \\ \mathbf{i}_2^T \\ \mathbf{i}_3^T \\ \mathbf{i}_4^T \\ \mathbf{j}_1^T \\ \mathbf{j}_2^T \\ \mathbf{j}_3^T \\ \mathbf{j}_4^T \end{bmatrix} \qquad (22)$$

and the partial shape matrix

$$S_{3\times3} = [\ \mathbf{s}'_1 \quad \mathbf{s}'_2 \quad \mathbf{s}'_3\ ] \qquad (23)$$

such that

$$W_{8\times 3} = RS'_{3\times 3} + \mathbf{t}'\mathbf{e}_3^T \ .$$

The primes here signal again that coordinates refer to the centroid of only the first three points. Then, this partial solution can be extended to $\mathbf{s}'_4$ by solving the following overconstrained system of six equations in the three unknown entries of $\mathbf{s}'_4$:

$$\begin{bmatrix} \mathbf{i}_1^T \\ \mathbf{i}_2^T \\ \mathbf{i}_3^T \\ \mathbf{j}_1^T \\ \mathbf{j}_2^T \\ \mathbf{j}_3^T \end{bmatrix} \mathbf{s}'_4 + \begin{bmatrix} a'_1 \\ a'_2 \\ a'_3 \\ b'_1 \\ b'_2 \\ b'_3 \end{bmatrix} = \begin{bmatrix} u'_{14} \\ u'_{24} \\ u'_{34} \\ v'_{14} \\ v'_{24} \\ v'_{34} \end{bmatrix} \qquad (24)$$

where

$$\begin{matrix} u'_{f4} = u_{f4} - a'_f \\ v'_{f4} = v_{f4} - b'_f \end{matrix} \quad \text{for} \quad f = 1,2,3 \ .$$

The matrix S' of the "primed" shape coordinates can now be registered with respect to their centroid to yield the "unprimed" coordinates:

$$\mathbf{s}_p = \mathbf{s}'_p - \frac{1}{4}S'\mathbf{e}_4 \quad \text{for} \quad p = 1,2,3,4$$

and the "unprimed" translation can again be found from equation (20).

In conclusion, the full motion and shape solution can be found in either of the following ways:

1. factor $W_{6\times 4}$ to find a partial motion and full shape solution, and propagate it to include motion for the remaining frame (equations (19));
2. factor $W_{8\times 3}$ to find a full motion and partial shape solution, and propagate it to include the remaining feature point (equation (24)).

4.2 Solution in the Presence of Noise

The solution propagation method introduced in the previous subsection can be extended to $2F \times P$ measurement matrices with $F \geq 4$ and $P \geq 4$. In fact, the only difference is that the propagation equations (19) and (24) now become overconstrained.

If the measurement matrix W is noisy, this redundancy is beneficial, since equations (19) and (24) can be solved in the Least Square Error sense, and the effect of noise is reduced.

In the general case of a noisy $2F \times P$ matrix W the solution propagation method can be summarized as follows. A possibly large, full subblock of W is first decomposed by factorization. Then, this initial solution is grown one row or one column at a time by solving systems analogous to those in (19) or (24), in the LSE sense.

However, because of noise, the order in which the rows and columns of W are incorporated into the solution can affect the exact values of the final motion and shape solution. Consequently, once the solution has been propagated to the entire measurement matrix W, it may be necessary to refine the results with a steepest-descent minimization of the residue

$$\|W - RS - \frac{1}{P}\mathbf{t}\mathbf{e}_P^T\|$$

(see equation (7)).

There remain the two problems of how to choose the initial full subblock to which factorization is applied and in what order to grow the solution. In fact, however, because of the final refinement step, neither choice is critical as long as the initial matrix is large enough to yield a good starting point. We illustrate this point in the first experiment of the next section.

5 More Experiments

In this section we first test the propagation method for occlusions with a laboratory experiment (subsection 5.1). Then, we demonstrate the robustness of the factorization method with two streams taken outdoors with a hand-held amateur camera. In the stream of subsection 5.2, occlusions are negligible. In the stream of subsection 5.3, they are dominant.

5.1 A Lab Experiment with Occlusions

In this image stream, a ping-pong ball with black dots marked on its surface is rotated 450 degrees in front of the camera. The rotation between adjacent frames is 2 degrees, so the stream is 226 frames long. Figure 11 shows the first frame of the stream, with the automatically selected features overlaid.

Every 30 frames (60 degrees) of rotation, the feature tracker looks for new features. In this way, features that disappear on one side around the ball are replaced by new ones that appear on the other side. Figure 14 shows the tracks of 60 features, randomly chosen among the 829 found by the selector.

If all measurements are collected into the noisy measurement matrix W, the two halves U and V of W have the same fill pattern: if the x coordinate of a measurement is known, so is the y coordinate. Figure 12 shows the *fill matrix* for our experiment. This matrix has the same size as either U or V, that is, $F \times P$. Every column corresponds to a feature point, and every row to a frame. Shaded regions denote known entries. The fill matrix has $226 \times 829 = 187354$ entries. Of these, 30185 (about 16 percent) are known.

To start the motion and shape computation, the algorithm first looks for a large, full submatrix. Rows and columns in the submatrix need not be adjacent. Finding a maximally large submatrix is both hard and unnecessary, since the initial matrix need only be large enough to allow for a reliable initialization of the motion and shape matrices.

Our system looks for the initial matrix by exploring the feature columns that originate in the first frame. In our experiment, there are 183 such features, which occupy the left 226×183 submatrix of the fill matrix of figure 12.

The matrix initialization routine sorts the columns in that submatrix in decreasing order of length (number of known entries). The routine then adds one column at a time. At every step, the size of the largest full matrix that can be formed with the given columns is equal to the number of columns times the length of the shortest column. The routine stops when adding an extra column would reduce the size of the largest full matrix.

In the example we consider, the initial submatrix happens to be square, of size 53×53. Motion and shape are computed for this submatrix with the factorization method of section 2. From this initial estimate, the solution is grown by repeatedly solving systems like those of equations (19), to add new rows, and of equation (24), to add new columns. Instead of 3×3 and 6×3, however, the systems are now more generally $3 \times P_n$ and $2F_n \times 3$ at step n.

The greater F_n and P_n, the more overconstrained the solution of each system. Consequently, we again use a greedy strategy to grow the initial matrix into the final solution. At each step of the growth, the row or column is chosen that has the maximum number of known entries in the same columns or rows as the current solution.

Eventually, all of the motion and shape values are determined. As a result, the unknown 84 percent of the measurement matrix can be hallucinated from the known 16 percent, with the only exception of the very few rows and columns containing fewer than three entries, for which no reconstruction is possible.

Figure 13 shows two views of the final shape results, taken from the top and from the side. Notice the missing features at the bottom of the ball in the side view. That is the part of the ball that was always invisible, because it rested on the rotating platform.

To display the motion results, we look at the $\mathbf{i}_f$ and $\mathbf{j}_f$ vectors directly. We recall that these unit vectors point along the rows and columns of the image sensor for frame f in $1, \ldots, F$. Because the ping-pong ball rotates around a fixed axis, both $\mathbf{i}_f$ and $\mathbf{j}_f$ should sweep a cone in space, as shown in Figure 15. The tips of $\mathbf{i}_f$ and $\mathbf{j}_f$ should describe two circles in space, centered along the axis of rotation. Figure 16 shows two views of these vector tips, from the top and from the side. Notice the double arc in the top part of figure 16. If the motion reconstruction were perfect, the two arcs would be indistinguishable.

5.2 An Outdoor Sequence

In this subsection, we describe the results of the factorization method on a stream of a real building, taken with a hand-held camera.

Outdoor images are harder to process than streams produced in the lab because lighting changes less predictably and the motion of the camera is more difficult to control. As a consequence, features are harder to track: the images are unpredictably blurred by motion, and corrupted by vibrations of the recorder's head, both during recording and digitization. Furthermore, the camera jumps and jerks produce a wide range of image disparities.

Figure 17 shows some of the 180 frames of the building stream. The overall motion covers a relatively small rotation angle, approximately 15 degrees.

The features found by the selection algorithm in the first frame are shown in figure 18. There are many false features. The reflections in the window partially visible in the top left of the image move non-rigidly, and so do the intersections between the roof and the horizontal edges of the siding on the right of the same window. A few more false features can be found in the lower left corner of the picture, where the vertical bars of the handrail intersect the horizontal edges of the bricks of the wall behind. We removed these bad features by hand, by masking away the two parts of the image involved. The front view of the shape results in Figure 19 clearly shows the cuts.

Figure 20 shows the tracks of 60 features selected randomly out of the 376 found by the selection algorithm. Notice the very jagged trajectories due to the erratic motion of the hand-held camera.

Figures 19 and 22 show a front and a top view of the building as reconstructed by the shape and motion recovery algorithm. To obtain these figures, we triangulated the tracked feature points, and mapped the pixel values in the first frame onto the resulting surface. The structure of the visible part of the building's three walls has clearly been reconstructed. In these figures, the left wall appears to bend on the right, where it intersects the middle wall. This occurred because the feature selector found features along the shadow of the roof just on the right of the intersection of the two walls, rather than at the intersection itself. Thus, the appearance of a bending wall is an artifact of the triangulation done for rendering.

Figure 21 shows two views of the $\mathbf{i}_f$ and $\mathbf{j}_f$ rotation vectors, in the style of figure 16. Notice how much more jagged the trajectories are, when compared with those of figures 16.

From the experiment described in this subsection we conclude that our approach works also for image streams taken outdoors with the jerky motion produced by a hand-held camera.

The identification of false features, that is, of features that do not move rigidly with respect of the environment, remains an open problem that must be solved for a fully autonomous system.

5.3 An Indoor Sequence with Occlusion

In the previous experiment we only tested the factorization method and ignored features that disappeared during the stream. In this subsection we describe an experiment where occlusion is a dominant phenomenon: a hand holds a cup and rotates it in front of the camera by about ninety degrees. Figure 23 shows four out of the 240 frames of the stream.

An additional difficulty in this experiment is the need for figure/ground segmentation. Since the camera was mounted on a fixed stand, however, this problem is easily solved: features that do not move belong to the background.

The stream presented in this experiment shows some nonrigid motion: as the hand turns, the configuration and relative position of the fingers changes slightly. This effect, however, is small and did not affect the results appreciably.

A total of 207 features was selected. Occlusions are marked by hand in this experiment. The fill matrix of figure 24 illustrates the occlusion pattern. Figure 26 shows the image trajectory of 60 randomly selected features.

Figures 25 and 28 show a front and a top view of the

cup and the visible fingers as reconstructed by the propagation method of section 4. These figures were obtained, as for the experiment in the previous subsection, by triangulating the tracked feature points, and mapping pixel values onto the resulting surface. The shape of the cup was recovered, as well as the rough shape of the fingers.

Figure 27 shows two views of the $\mathbf{i}_f$ and $\mathbf{j}_f$ rotation vectors. Notice that motion is smoother here than for the hand-held camera stream (figure 21), but is still far less regular than in the laboratory experiments (compare with figure 16).

From the experiment discussed in this subsection, we conclude that the solution propagation method of subsection 4 deals well with occlusion also when the image stream is taken outdoors with poorly controlled motion and lighting. The most urgent problem to be solved to make this propagation method into a fully autonomous system is how to detect occlusion events in the image stream.

6 Conclusion

The factorization method is a first step towards a flexible and robust method for shape and motion recovery from television images. Even now, in its early stages of development, the method works for actual image streams taken outdoors with an inexpensive camera under erratic motion.

A posteriori, the conceptual simplicity of the factorization method is perhaps its most surprising feature. But this very simplicity makes the method extensible to more and more general situations. The treatment of occlusions is a case in point.

References

[Adi85] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE PAMI*, 7:384–401, 1985.

[BBM87] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *IJCV*, 1(1):7–55, 1987.

[BCC90] T.J. Broida, S. Chandrashekhar, and R. Chellappa. Recursive 3-d motion estimation from a monocular image sequence. *IEEE AES*, 26(4):639–656, July 1990.

[BH83] A. R. Bruss and B. K. P. Horn. Passive navigation. *CGVIP*, 21:3–20, 1983.

[BJT90] J. L. Barron, A. D. Jepson, and J. K. Tsotsos. The feasibility of motion and structure from noisy time-varying image velocity information. *IJCV*, 5(3):239–269, 1990.

[DA90] C. H. Debrunner and N. Ahuja. A direct data approximation based motion estimation algorithm. In *Proc. of the 10th Intl. Conf. on Pattern Recognition*, pages 384–389, Atlantic City, NJ, June 1990.

[GR71] G. H. Golub and C. Reinsch. *Singular Value Decomposition and Least Squares Solutions*, volume 2, chapter I/10, pages 134–151. Springer Verlag, New York, NY, 1971.

[Hee89] J. Heel. Dynamic motion vision. In *DARPA IU Workshop*, pages 702–713, Palo Alto, Ca, May 23-26 1989.

[HHN88] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, 5(7):1127–1135, July 1988.

[HJ89] D. J. Heeger and A. Jepson. Visual perception of three-dimensional motion. Tech. Report 124, MIT Media Laboratory, Cambridge, Ma, December 1989.

[LK81] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *7th IJCAI*, 1981.

[MKS89] L. Matthies, T. Kanade, and R. Szeliski. Kalman filter-based algorithms for estimating depth from image sequences. *IJCV*, 3(3):209–236, September 1989.

[Pra80] K. Prazdny. Egomotion and relative depth from optical flow. *Biological Cybernetics*, 102:87–102, 1980.

[RA79] J. W. Roach and J. K. Aggarwal. Computer tracking of objects moving in space. *IEEE PAMI*, PAMI-1(2):127–135, April 1979.

[SA89] M. E. Spetsakis and J. (Yiannis) Aloimonos. Optimal motion estimation. In *IEEE Workshop on Visual Motion*, pages 229–237, Irvine, California, March 1989.

[TH84] R. Y. Tsai and T. S. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE PAMI*, PAMI-6(1):13–27, January 1984.

[TK90] C. Tomasi and T. Kanade. Shape and motion without depth. In *Third ICCV*, Osaka, Japan, December 1990.

[TK91a] C. Tomasi and T. Kanade. Shape and motion from image streams: a factorization method - 2. point features in 3d motion. Tech. Report CMU-CS-91-105, Carnegie Mellon University, Pittsburgh, PA, January 1991.

[TK91b] C. Tomasi and T. Kanade. Shape and motion from image streams: a factorization method - 3. detection and tracking of point features. Tech. Report CMU-CS-91-132, Carnegie Mellon University, Pittsburgh, PA, April 1991.

[Ull79] S. Ullman. *The Interpretation of Visual Motion*. The MIT Press, Cambridge, Ma, 1979.

[WW85] A. M. Waxman and K. Wohn. Contour evolution, neighborhood deformation, and global image flow: planar surfaces in motion. *Intl. Journal of Robotics Research*, 4:95–108, 1985.

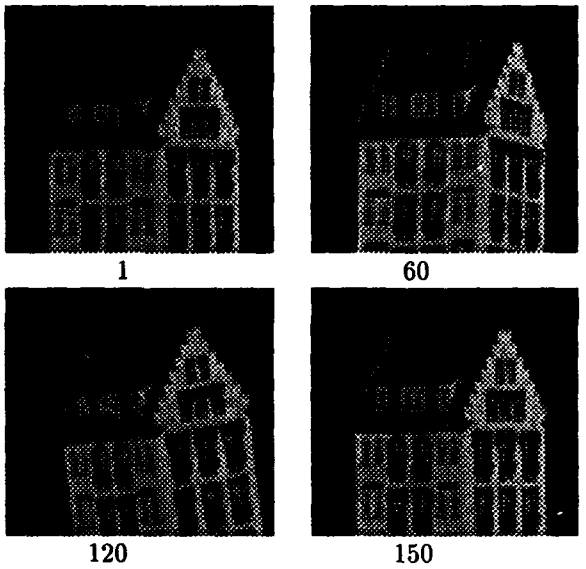


Figure 3: Some frames in the sequence. The whole sequence is 150 frames.

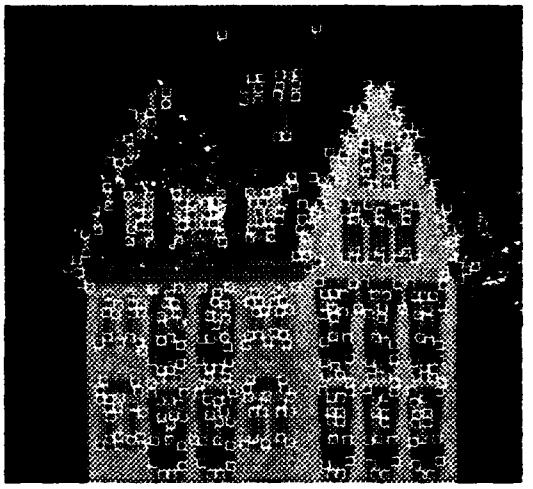


Figure 5: The 430 features selected by the automatic detection method.

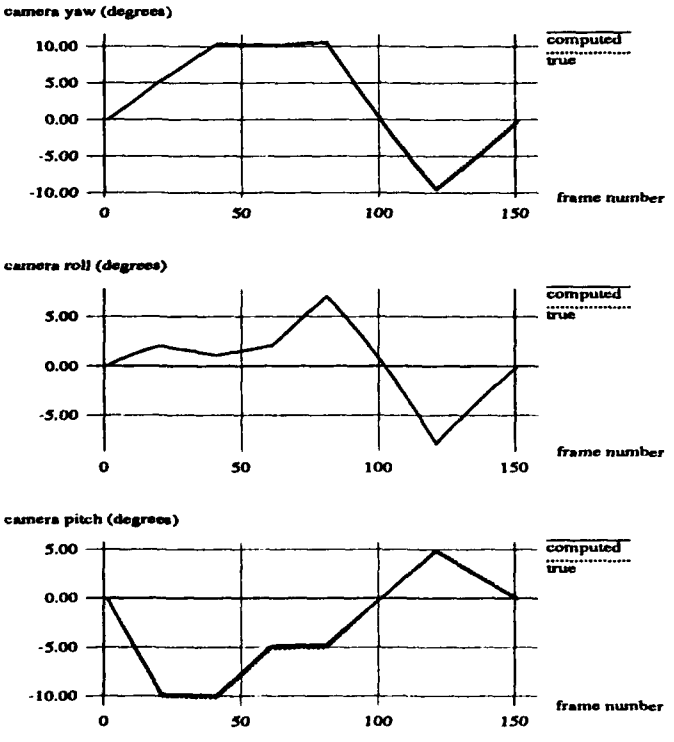


Figure 4: True and computed camera yaw, roll, pitch.

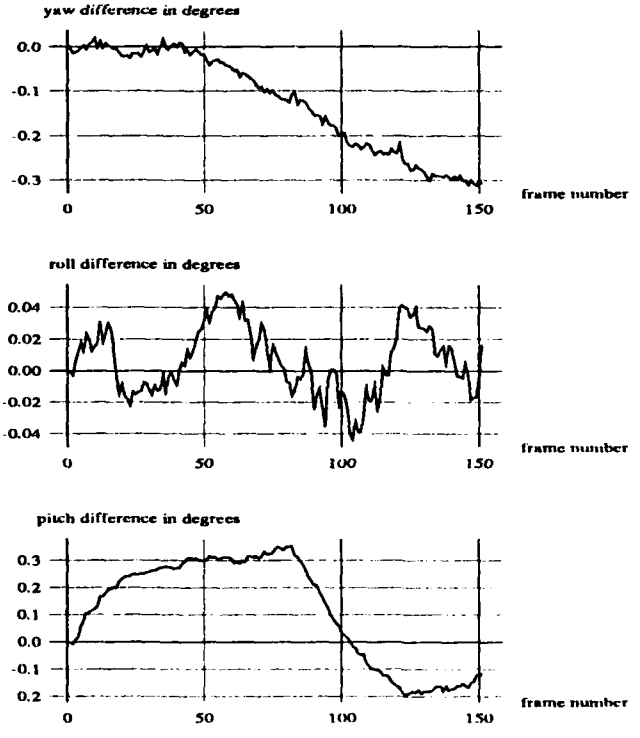


Figure 6: Blow-up of the errors in figure 4.

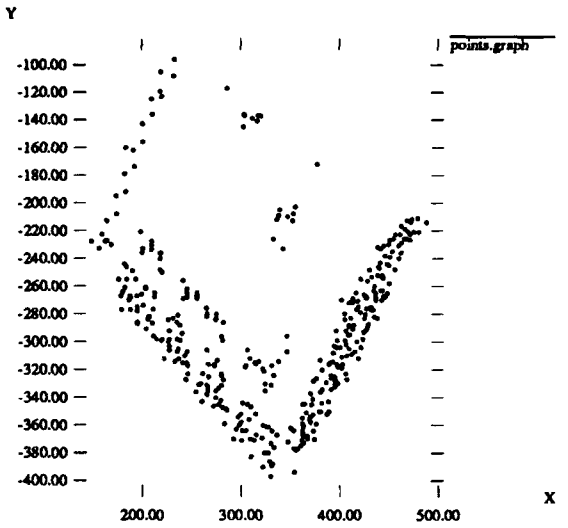


Figure 7: A view of the computed shape from approximately above the building (compare with figure 8).

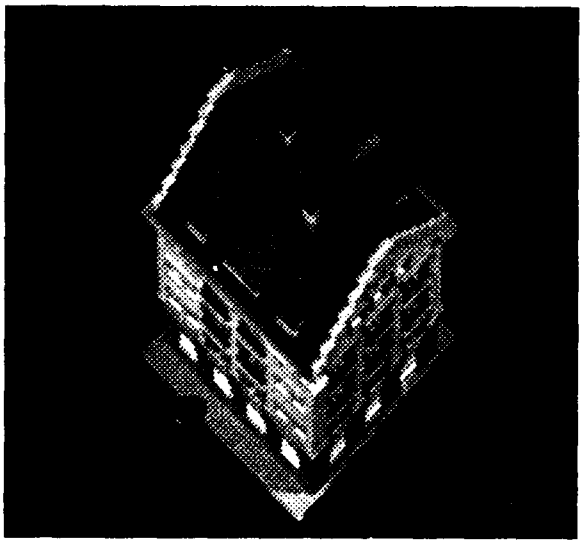


Figure 8: A real picture from above the building, similar to figure 7.

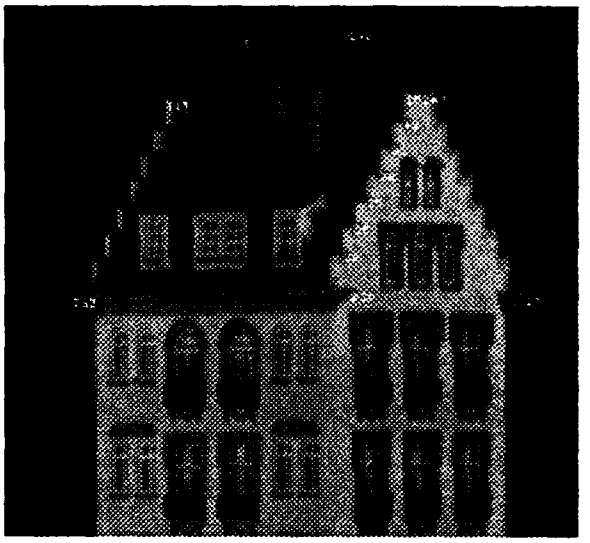


Figure 9: For a quantitative evaluation, distances between the features shown in the picture were measured on the actual model, and compared with the computed results. The comparison is shown in figure 10.

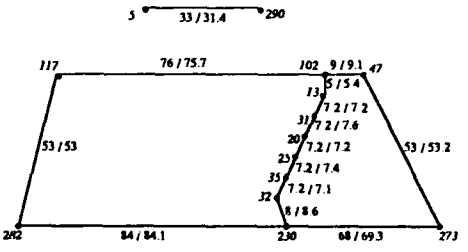


Figure 10: Comparison between measured and computed distances for the features in figure 9. The number before the slash is the measured distance, the one after is the computed distance. Lengths are in millimeters. Computed distances were scaled so that the computed distance between features 117 and 282 is the same as the measured distance.

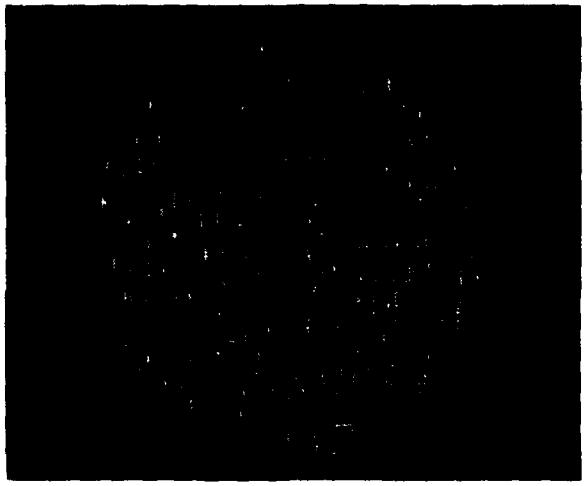


Figure 11: The first frame of the ping-pong stream, with overlaid features.

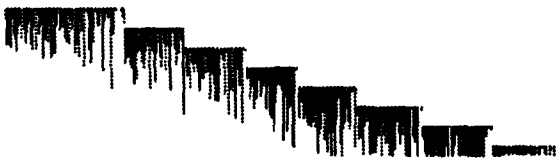


Figure 12: The fill matrix for the ping-pong ball experiment. Shaded entries are known.

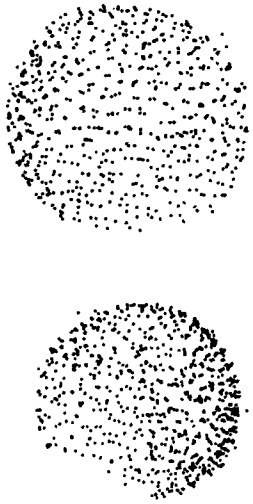


Figure 13: Top and side views of the reconstructed ping-pong ball.

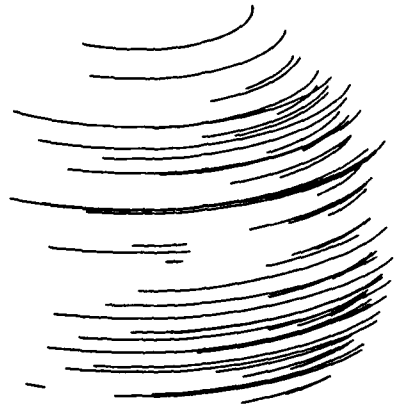


Figure 14: Tracks of 60 randomly selected features from the stream of figure 11.

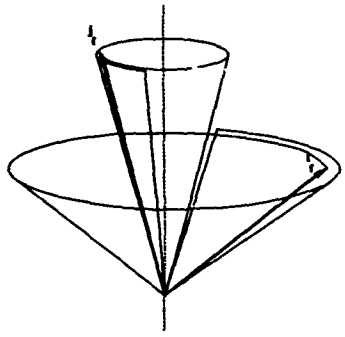


Figure 15: Rotational component of the camera motion for the ping-pong stream. Because rotation occurs around a fixed axis, the two mutually orthogonal unit vectors $\mathbf{i}_f$ and $\mathbf{j}_f$, pointing along rows and columns of the image sensor, sweep two 450-degree cones in space.

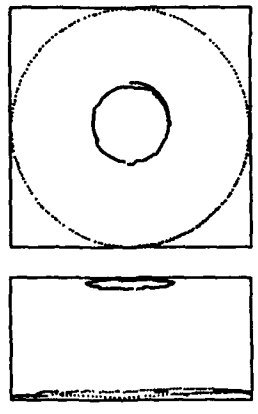


Figure 16: Top and side views of the $\mathbf{i}_f$ and $\mathbf{j}_f$ vectors identifying the camera rotation. See Figure 15.

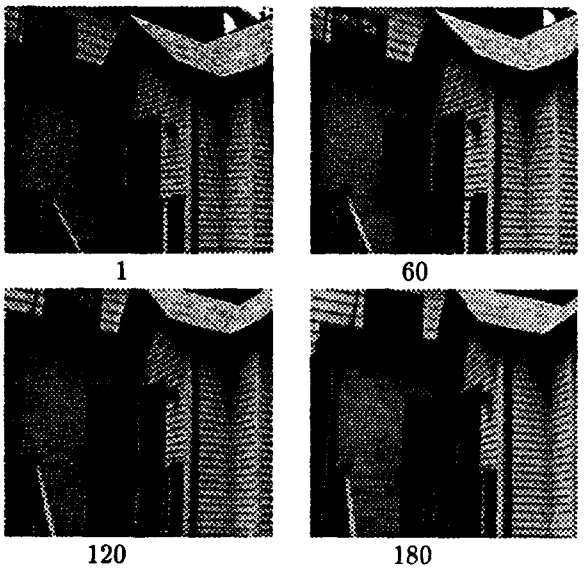


Figure 17: Four out of the 180 frames of the real house image stream.



Figure 18: The features selected in the first frame of the real house stream (figure 17)

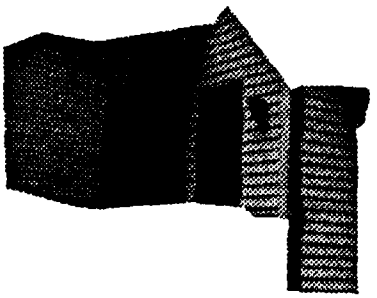


Figure 19: A front view of the three reconstructed walls, with the original image intensities mapped onto the resulting surface.



Figure 20: Tracks of 60 randomly selected features from the real house stream (figure 17.)

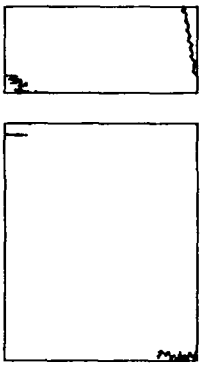


Figure 21: Top and side views of the $\mathbf{i}_f$ and $\mathbf{j}_f$ vectors identifying the camera rotation for the real house stream.

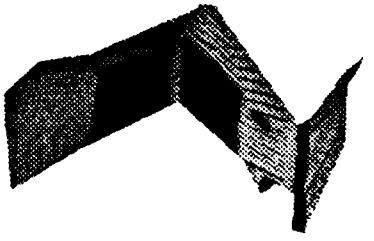


Figure 22: A view from above of the three reconstructed walls, with image intensities mapped onto the surface.

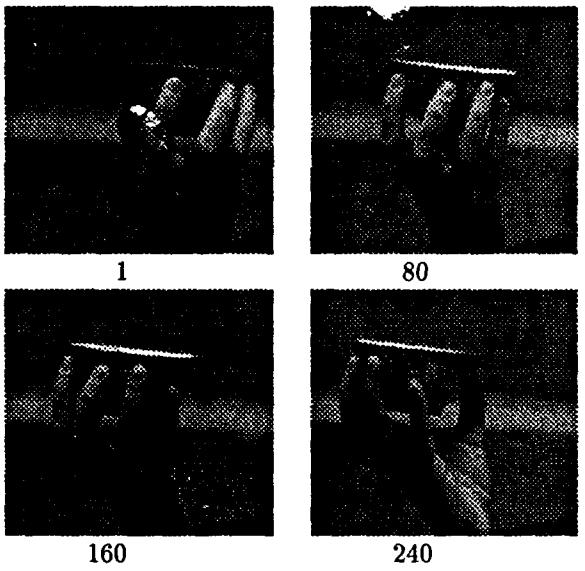


Figure 23: Four out of the 240 frames of the cup image stream.

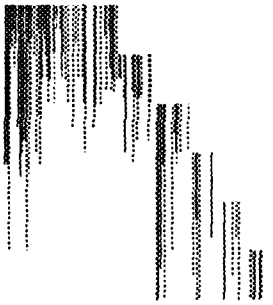


Figure 24: The 240 × 207 fill matrix for the cup stream (figure 23). Shaded entries are known.



Figure 25: A front view of the cup and fingers, with the original image intensities mapped onto the resulting surface.



Figure 26: Tracks of 60 randomly selected features from the cup stream.

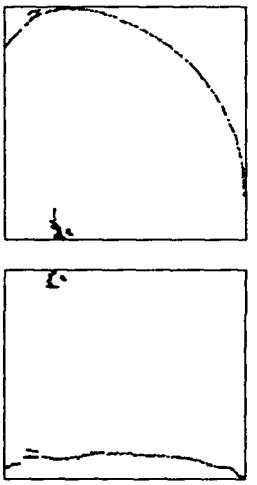


Figure 27: Top and side views of the $\mathbf{i}_f$ and $\mathbf{j}_f$ vectors identifying the camera rotation for the cup stream (figure 23).

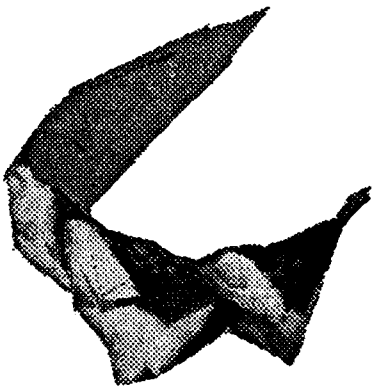


Figure 28: A view from above of the cup and fingers with image intensities mapped onto the surface.

Integrated 3D Recovery and Visualization of Flight Image Sequences

Sanghoon Sull and *Narendra Ahuja* *
Beckman Institute
University of Illinois, Urbana, IL 61801

Abstract

We present an algorithm for estimating and visualizing motion of an observer relative to a scene. A key feature of the approach presented is an integrated use of multiple image attributes which are shared by both estimation and visualization processes. We focus on flight image sequences, i.e., image sequences acquired by an observer moving smoothly over a planar, textured surface. The approach presented allows the use of image cues such as regions, point features, optical flow, texture gradient and vanishing line. The integration of information in these diverse cues is carried out using optimization. Visualization is done using the image attributes extracted from the image sequence during 3D recovery. For reliable estimation, a sequential batch method is used to compute motion and structure. Experimental results on motion and structure estimation and visualization are presented for a real image sequence digitized from a commercially available video tape. The visualization sequence appears very similar to the original sequence in informal viewing on a workstation monitor.

1 Introduction

This paper is concerned with the problem of estimating and visualizing motion and structure of a scene as seen by an observer in relative motion. There are two objectives of the paper. First, it addresses the problem of recovering motion and structure parameters from a monocular image sequence. Second, it uses the intermediate results of the recovery process to sythesize an image sequence that depicts the motion and structure. A key feature of the approach presented which helps meet both objectives is an integrated use of multiple image attributes or cues. These cues carry the motion and structure information of interest to different degrees, and have different, often complimentary, strengths and shortcomings. Thus when a given attribute does not contribute significantly to the estimation process, other, more pertinent cues help achieve reliable estimation. The goal is to estimate motion and structure parameters such that the estimated parameters best explain the presence of *all* of the observed image cues throughout the image sequence.

The result of the integrated recovery process is two fold: it simultaneously gives the estimates of the motion and structure parameters as well as identifies those image cues which are found to contribute to these estimates. In a sense, this amounts to the identification of image characteristics that carry the *most information* about the relative motion and structure. A result of this is that we can use these image attributes for creating depictions of the scene based on the premise that the display of these attributes will be the most cost-effective way of communicating to the observer the same motion and structure characteristics as perceived from the original image sequence. These depictions may thus also be viewed as a three-dimensional (3D) interpretation based approach to image sequence compression which obviously should have very high compression ratios.

The identification and analysis of the relative strengths of different cues for the problem at hand is a research problem in itself. In general, the available cues, and sometimes even their relative merits, depend upon the scene under consideration. Within the context of the navigation scenario as mentioned earlier, in this paper we focus on the problem of an observer moving above a planar, textured surface such as while in an aircraft which is landing or taking off. The goal is to recover the translational and rotational motion of the observer and the orientation of the plane as a function of time, from the sequence of images of the plane acquired during the motion. The approach we present allows the use of the following image cues: regions, point features, optical flow, texture gradient, and vanishing line. (The vanishing line is defined as the intersection of the image plane with a plane which includes the camera center and is parallel to the object plane.) These cues could be changed to achieve increased robustness for the given scenario or to suit a different scenario, without affecting the basic approach.

The framework for integration used in this paper is one of optimization. The objective function to be minimized represents the error between the observe image attributes and those corresponding to motion and structure parameters, structure consistency through the image sequence, and a measure of the nonsmooth-

*The support of DARPA and NSF under grant IRI-89-02728 is gratefully acknowledged.

ness of motion. The visualization is done using the attributes extracted from the image sequence during 3D recovery.

Section 2 presents an overview our algorithm for integrated 3D recovery and visualization of motion over a planar, textured surface from a monocular image sequence. Section 3 presents the results obtained in experiments with a sequence of 29 images, digitized from a commercially available videotape of a film taken from an aircraft. The visualization sequence obtained appears compellingly similar to the original when the two are played side by side on a SUN workstation monitor, although we have not performed any rigorous psychophysical experiments about this.

2 Algorithm

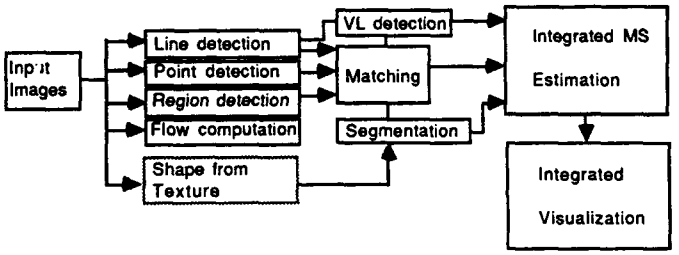


Figure 1: A framework for our approach.

There are six major steps in our approach as shown in Fig.1. The third and fourth step are represented by one block (Integrated Estimation) and the shape from texture was not used in this paper. Details are presented in [5].

The goal of the first step is to detect independently points, lines and regions in each frame. Optical flow is also computed between each pair of adjacent frames. Candidate vanishing lines are identified from among the detected lines. Images cues are considered only below the candidate vanishing lines to reduces computation time. The optical flow is used at only those locations where a point feature detector responds. This helps in selecting reliable flow since point features are usually detected at locations having high intensity gradients.

The second step establishes correspondences between features in each pair of adjacent images using a first order model of the image plane displacement of the features. Thus, the displacement vector (D_x, D_y), which represents the position after motion of a feature located at (x,y), is given by:

$$D_x = c_0 + c_1 x + c_2 y \quad (1)$$

$$D_y = c_5 + c_6 x + c_7 y. \quad (2)$$

Each distinct image plane motion is represented by a distinct set of values for the coefficients c_i's. All such sets of c_i's supported by the feature locations in two adjacent frames are identified using Hough transform. The support for any set of c_i values is computed from the image plane distances between the observed feature locations and those predicted by the c_i values in the set under consideration. The well supported c_i values then simultaneously determine a segmentation of the images into distinctly moving objects, and establish correspondences between features contained within each object. In general, correspondences are not found for all attributes contained in an object; some of the attributes remain unmatched.

In third step, the correspondences found are used to obtain the fits of the displacement vectors in different segments to a second order model. The errors in the resulting fits are used to merge any distinct first-order segments that now have identical second order parameters. In the problem at hand, this leads to a merger of all segments since all features are known to belong to a single plane. Here, the motion and structure parameters can be linearly computed from pairs of adjacent images.

The objective of the fourth step is to use the established feature correspondences to determine motion and structure (i.e., the orientation of the plane). These parameters could be computed in a linear fashion from pairs of adjacent images. However, when an image is paired with its predecessor and successor images in the sequence, the resulting structure parameters will in general not be identical. Thus, the requirement of such consistency of structure parameters must be explicitly enforced. This makes the motion and structure estimation a nonlinear problem. To enforce this requirement, we must consider a batch of frames at a time. We therefore perform motion and structure estimation over a sliding window of N frames along the image sequence. For each such window, the motion and structure parameters are estimated by minimizing the following objective function with respect to motion and surface orientation parameters:

$$G \stackrel{\text{def}}{=} \sum_{k=0}^{N-2} (\lambda_r E_{k,r} + \lambda_p E_{k,p} + \lambda_l E_{k,l} + \lambda_f E_{k,f} + \lambda_{sc} E_{k,sc} + \lambda_{mco} E_{k,mco}) + \sum_{k=0}^{N-1} \lambda_{vl} E_{k,vl} \quad (3)$$

where

$E_{k,r}$	:	error from regions
$E_{k,p}$	:	error from points
$E_{k,l}$	:	error from lines
$E_{k,f}$	:	error from optical flows
$E_{k,sc}$	:	structure constraints
$E_{k,mco}$	:	motion coarseness
$E_{k,vl}$	:	error from the vanishing line.

This objective function contains the contributions of multiple features to the scene characteristics to be estimated. Each contribution is weighted by a factor λ. The term $E_{k,sc}$ does not restrict motion nor structure. It requires that the structure at $t = k$ should be the same whether computed from the pair of frames $(k-1, k)$ or $(k, k+1)$. $E_{k,mco}$ makes the motion parameters vary smoothly which is quite reasonable especially when we are dealing with multiple frames. $E_{k,vl}$ is the penalty term which makes the orientation parameters stay within a certain range of the initial values computed from the candidate vanishing lines. We could use the linear solutions from the previous step as initial guesses for this nonlinear minimization, however, the surface orientations corresponding to the detected vanishing lines are used since they are usually better than the linear soultions. The objective function G is optimized with respect to surface orientation parameters only. Hence, the number of iteration variables is $2(N-1)$ for surface unit normals in a batch since motion parameters which minimize G are linearly computed for given surface orientations.

The motion and structure estimates obtained from each batch are compatible with the attributes of the images in the batch. Clearly the larger the batch, the more compatible the estimates will be with the image sequence. In this fifth step, motion parameters derived from batch computations are sequentially updated as follows although we can use any sequential updating algorithm available in the literature:

$$m(k) = m(k-1) + \mu_k(\hat{m}(k) - m(k-1)) \qquad (4)$$

where $\hat{m}(k)$ is the present estimate obtained using the batch approach on the most recent block of data at $t = k$, and μ_k is the smoothing parameter based on the objective function value at $t = k$. The result is a set of estimates of rotation and normalized translation parametrs obtained from the image sequence. Then, using these estimated motion parameters, we recompute the orientations for all frames. Finally, we rescale translation and structure parameters starting from the initial frame by using any set of points on the plane.

Finally, in the sixth step, we use the estimated motion and structure parameters for visualization. For this purpose, only those image attributes for which correspondences are found in the first step are considered. The remaining attributes are discarded. The synthesized sequence is a simplified depiction of the retained attributes. Now, an attribute may not be present throughout an image sequence because, for example, it may not be detected in each image. Each such attribute is introduced in each image where it is missing. This is done by extrapolating from the nearest frame where it is detected, using the estimated motion and structure values. Jacobian-based area relationship is used for each pixel in a region for extrapolation.

3 Experimental results

We derived a sequence of 29 frames from a commercially available VHS video tape of a film shot from a flying aircraft. The digitization was done with a resolution of 720 by 486. The frames were initially digitized in three RGB channels with 8 bits per color. Color images were converted to HSV (Hue, Saturation and Value) [3]. In our experiments, we use only the V component which is defined as $\max(R, G, B)$. The size of each image is reduced to to 600 by 464 to remove the jitter on the boundary. In Fig.6, we show the resulting 29 frames. Since the commercial VHS tape is far from having the quality of the mastertape, digitized images are very noisy. There is also blurring of images since they were take from a camera mounted on a flying aircraft.

Next, we extract regions, lines and flow as image attributes. Examples of these detected features are shown in Figs.2, 3 and 4. The results of segmentation, matching and merging for one frame are shown in Fig.5. For segmentation and matching, we use only regions and flow which are below the detected candidate vanishing lines.

When we iteratively minimize Eq.(3), we set λ_l and λ_p equal to zero since we do not use lines and point features. By setting $\lambda_{sm} = 0$, we do not impose any smoothness restriction on motion and structure. The values of λ_r and λ_f used are equal to one. Compared to them, we use larger values of λ_{sc} and λ_{vl}.

The visualization sequence (Fig.7) is synthesized by displaying (i) those regions whose correspondences are used for motion and structure estimation and (ii) the vanishing line derived from the estimated surface orientation. As explained earlier, missing regions are introduced based on the estimated motion and structure parameters. The original sequence and the resulting visualization sequence are presented in Fig.6 and Fig.7. If we watch the two sequences visually as they are played on a SUN workstation monitor, we perceive the same motion and structure from them in an informal viewing.

Acknowledgement

We would like to thank Robert Patterson at National Center for Supercomputer Application for his help in digitizing the images. Also, we thank Dr. Yuncai Liu, Xiaoping Hu, Darell Hougen and Taehoon Cho for letting us use their programs for line detection, point feature detection, flow computation and region detection, respectively.

References

[1] G. Adiv, "Determining three-dimensional motion and structure from optical flow generated by several moving objects," *IEEE Trans. Pattern Anal. Machine Intell.*, vol.PAMI-7, no.4, July, 1985

[2] J. Rolfe and K. Staples *Flight Simulation*, Cambridge University Press, 1986

[3] D. Hearn and M. Baker *Computer Graphics*, Prentice-Hall, 1986

[4] S. Sull and N. Ahuja, "Segmentation, Matching and Estimation of Structure and Motion of Textured Piecewise Planar Surfaces," in *Proc of IEEE Workshop on Visual Motion*, Princeton, NJ, Oct., 1991, pp.274-279

[5] S. Sull and N. Ahuja, "Integration of Multiple Cues for 3D Recovery and Visualization of Flight Image Sequences," submitted to *IEEE Trans. Pattern Anal. Machine Intell.*

[6] Y. Liu, *Rigid Object Motion Estimation from Intensity Images Using Straight Line Correspondences.* Ph.D dissertation, University of Illinois, Urbana-Champaign, 1990



Figure 2: Extracted regions at $t = 25$.

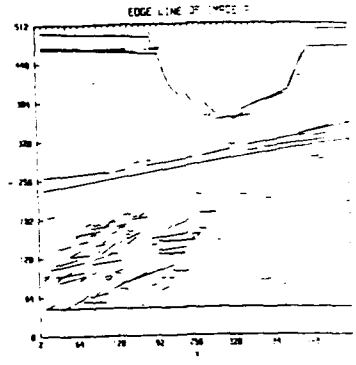


Figure 3: Extracted lines at $t = 25$.



Figure 4: Computed flow between $t = 25$ and 26.



Figure 5: Segmented regions and optical flow between $t = 25$ and 26.

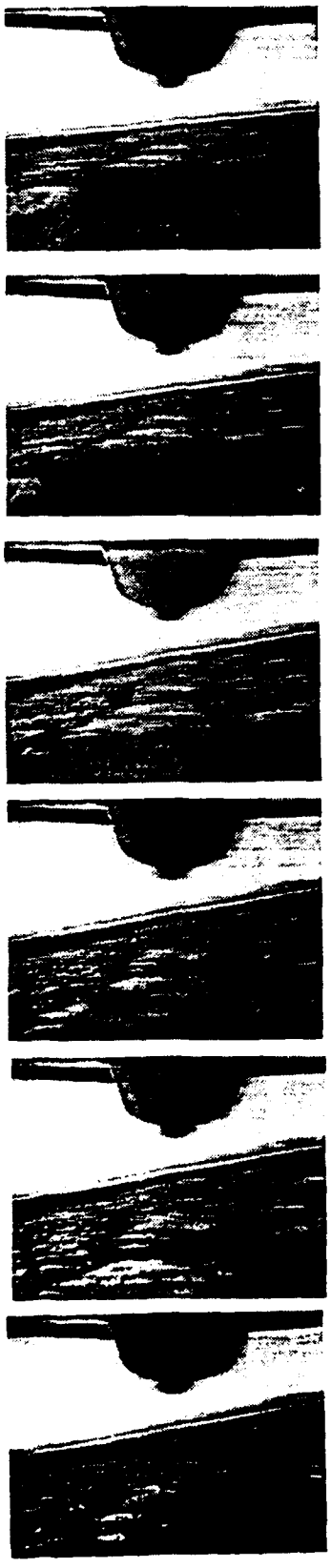


Figure 6: Input images sequence used in our experiments

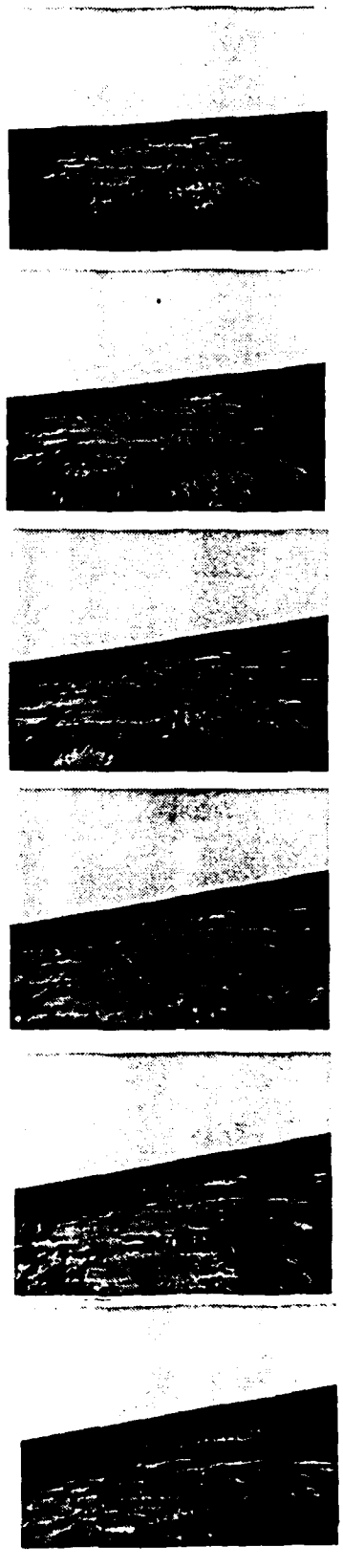


Figure 7: Visualization image sequence

Refinement of Noisy Correspondence Using Feedback from 3-D Motion *

Yong Cheol Kim and Keith Price
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273

Abstract

We present an approach to improve the matching of features in the domain of feature-based motion analysis for multiple frames. In an automated system, correspondence data are usually noisy and fragmented. Either synthetic data with Gaussian noise added or manually selected feature correspondence has been most commonly used for motion analysis. We combined establishment of correspondence and motion estimation and developed a technique that gradually refines the initial noisy correspondence data and links fragments of a single feature into one trajectory using feedback from the prior 3-D motion estimation. First, 3-D motion parameters are estimated using the noisy initial correspondence data. Then, each noisy trajectory is partitioned into overlapping subsets each of which conforms to the estimated motion. The largest set is selected as the input to the next motion estimation. This selection process is repeated and the gaps in the refined correspondence data are filled by guidance from the predicted motion. Test results for a standard real image sequence are presented.

1 Introduction

Motion analysis is one of the important research areas in computer vision. The goal of motion analysis is to detect coherently moving objects and to recover the relative motion between a viewer and the objects as well as the structure of the environment. The information about moving objects may assist solving problems such as segmentation and shape analysis.

Feature-based motion analysis techniques are built on a common framework - feature extraction, establishment of correspondence, estimation of motion parameters and recovery of 3-D structure. The recovered structure is a sparse depth map and interpolation is needed if dense depth map is required.

Most earlier motion analysis efforts have been concerned with only two or three frames [Roach and Aggarwal, 1980; Tsai and Huang, 1984; Adiv, 1985; Anandan, 1987]. Reconstruction of 3-D structure from 2-D data is an inherently ill-posed problem. The formulations are very sensitive to noise in the input data and the results are often unstable. To overcome the inherent instability, longer sequence of images must be used.

Broida [Broida *et al.*, 1990] analyzed images of objects undergoing a general motion, assuming no knowledge of the object. They also developed Cramer-Rao lower bounds, which indicates a theoretical bound for the performance of the estimation. Shariat [Shariat and Price, 1990] presented a method for the computation of 3-D motion parameters by using the correspondence of point features over multiple frames. They developed a mathematical formulation for 3 points in 3 frames, 2 points in 4 frames and 1 point in 5 frames.

In order to fully utilize the advantage of multiple frames, feature correspondences ranging over the entire sequence are desired. However, obtaining reliable correspondence over multiple frames is a nontrivial task for several reasons.

First, the set of extracted features is not identical for all frames. This can be caused by variations in the extractions of features. Therefore, a feature may not have a proper correspondence and the number of features may be different from frame to frame. If a feature fails to be extracted in some of the frames due to occlusion or other reasons, its trajectory is fragmented into shorter ones and the depth estimation of the feature gets less stable. These short corresponding sequences should be linked into one for a reliable depth estimation. Second, a correct correspondence over the entire sequence requires correct matching over all pairs of adjacent frames. Once a matching error occurs between any two frames, the entire trajectory may be useless.

Jenkin [Jenkin, 1983] presented a method for tracking the 3-D motion of points from their 2-D perspective images as viewed by a binocular vision system. He used a smoothness assumption that the location, the speed and the direction of a given point feature would be relatively unchanged from one frame to the next.

*This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-90-C-0078. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

Sethi [Sethi and Jain, 1987] suggested a method of establishing the correspondence based on the smoothness of motion. Their work used an assumption that the number of extracted points are always the same except for one frame where some points may disappear due to occlusion. A trajectory is generated for each point by a nearest neighbor heuristic. Then, the matches of intermediate points are updated until there is no further gain of smoothness.

Cheng [Cheng and Aggarwal, 1990] relaxed the restriction of the same number of points for every frame. The first stage of their algorithm establishes an initial correspondence by maximizing the smoothness of motion. The second stage inspects the last four frames and arranges the correspondence among them.

The test results for the last three are either on synthetic data or on manually selected point features from real images. Fletcher [Fletcher *et al.*, 1991] developed an algorithm for tracking multiple feature points in a real image sequence. After an initial correspondence of the first few frames, the matching for subsequent frames is guided by the 2-D motion of the features. The tracking is highly dependent on the prediction from the 2-D motion and fails when the prediction fails.

Recently, there has been work on automated motion analysis systems. Leung [Leung *et al.*, 1991] combined feature extraction, matching and motion estimation algorithms, and applied it to well-calibrated outdoor scenes. Chandrashekhar [Chandrashekhar and Chellappa, 1991] also combined the same operations and used the prediction from 3-D motion for feature correspondence. Sull [Sull and Ahuja, 1991] developed an algorithm which segments and matches the regions and then estimates 3-D motion parameters and structure from two views.

Analysis of motion from a real image sequence is a complex task. A more reliable result can be achieved in an integrated approach, where feature extraction, establishment of correspondence and motion analysis are performed in cooperative manner, exchanging information among the separate sub systems. For example, the performance of feature matching is enhanced when guided by prediction from motion. The feature extraction of an object in one frame can be guided by its expected properties induced from the corresponding objects in other frames. We show that an improvement can be achieved with an integrated approach. We combined feature correspondence and motion estimation. Initial noisy correspondence data are gradually refined and fragments of a trajectory of a single feature are linked into one using feedback from the 3-D motion estimation.

This paper describes part of a system that extracts the structure of a scene from a moving camera. The motion of the camera is arbitrary, but in applications such as autonomous vehicles, the motion would be roughly along the line of sight of the camera. This work is in the domain of feature-based motion analysis. The work reported here concentrates on the use of 3-D motion to improve the matching of features and the resulting improvement in the motion and structure estimation.

In the next section, we give an overall description of the system. A detailed description of the refinement of noisy correspondence using 3-D motion information is in section 3. Finally, we present the test results on real image data, followed by a brief conclusion.

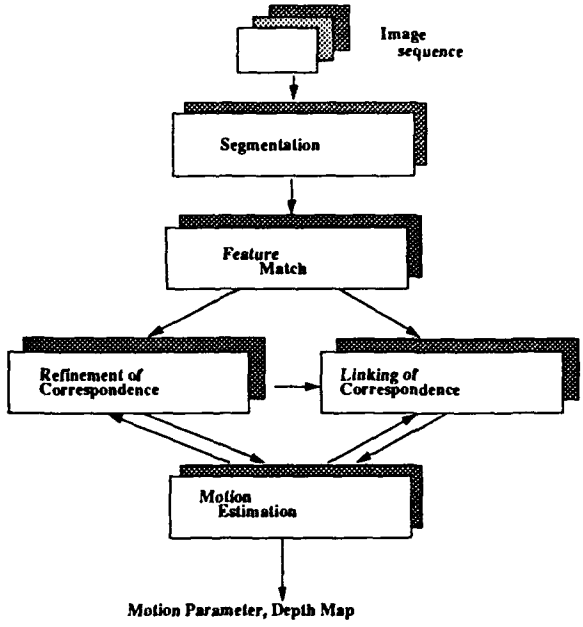


Figure 1: Overall Block Diagram

2 Description of Integrated System

The block diagram in figure 1 shows the use of each subsystem and the feedback of information. The basic features are regions and corners extracted from the contour of the regions. Each block is described briefly.

Each image in the sequence is segmented into a set of regions by a recursive splitting technique that uses the statistics of image attributes [Ohlander *et al.*, 1978]. Segmented regions are recursively segmented into smaller regions until the size of the region is too small or the attribute is inseparable. In our work, the segmentation of the current frame is guided by that of the previous one for a consistent segmentation of regions from frame to frame.

Corners are obtained at the intersection of those linear segments which approximate the contour of a region. When an object is clear polygonal shape, the position of the corners tend to be consistent for a wide range of scale changes and rotation. Contours of natural objects such as trees, rocks and colored spots on the ground are usually irregular curves and it is less likely to obtain long matches of consistent corners of natural objects.

Feature matching is done in a hierarchical way to speed up computation and increase stability. Region correspondences are first established and corners are matched along the matched regions. Matching is performed both in *forward* and *backward* direction to ensure one-to-one match of features.

Relaxation-based symbolic matching [Faugeras and Price, 1981] is used for both region and corner matching. The matching system uses feature-based symbolic description. For region matching, the features include average values of the image intensity, size, location and simple shape measures. Relations included in the description are also those which are easily computed, such as adjacency, relative position and near-by.

For corner matching, the properties of the corners are position, inside-angle, in and out-direction (the angle and the orientations of the two line segments forming the corner), radial-angle and radial-distance (the orientation and distance from the center of the region).

3-D trajectory and structure are estimated with chronogeneous motion analysis [Franzen, 1991], which can handle uniform acceleration with constant translation and rotation. This program requires at least 3 frames with visible points but allows points to be skipped in some frames.

3 Refinement and Linking of Correspondence

The main sources of noise in correspondence data are inconsistent feature extraction and erroneous feature match . These two are somewhat inseparable since many of the matches among the inconsistent features are correct in the symbolic sense while the resulting correspondence data are noisy. When the object has a smooth boundary, the extracted contour has no distinct corners when approximated by linear segments and the location of a corner may drift from frame to frame. This uncertainty in the position degrades the reliability of the estimation of the motion parameters. One advantage of two-step feature match (region and corner) is that, when the underlying region match is entirely at fault, very few corner matches (whether they are correct or incorrect) are obtained since the similarity and compatibility between corners are very low in the relaxational matching.

Image acquisition under unfavorable conditions also deteriorates the quality of the correspondence. When the camera is mounted on a vehicle moving on terrain, the motion of the camera may have fluctuations, which violates the assumption that the 3-D motion parameters are constant throughout the observation time. When the motion of the camera is a little out of the normal path, all the features in that frame are uniformly shifted. Uniform shifts causes correlated noise, which is known to have worse effects than random noise.

It is desirable to detect and delete the erroneous part of noisy data and select only the good part for motion estimation. We review two measures of reliability for the refinement, smoothness and fitting error.

The smoothness of the curve formed by the trajectory proved to be useful 2-D information for the establishment of correspondence [Sethi and Jain, 1987] when no knowledge about the motion is available. The usefulness of smoothness as the measure of reliability of established correspondence, however, is limited in our work. First, the trajectories are not always long enough to produce any meaningful smoothness measure. Furthermore, the

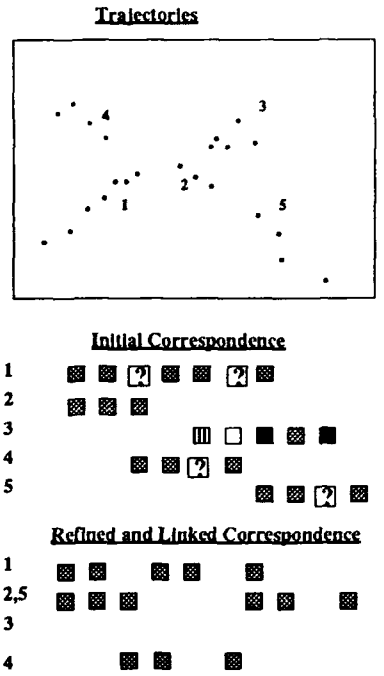


Figure 2: Refinement and linking and noisy trajectories

smoothness of a trajectory with just one or two spurious points can be very low while an entirely faulty trajectory may form a very smooth trajectory.

The fitting error is the difference between the input position and its estimated position when the motion information is available. This measure can indicate the quality of a trajectory, but does not point out which points are the spurious ones (gross points) in the trajectory [Fischler and Bolles, 1981] when the motion estimation is based on least mean square error method.

When one or two spurious points cause a trajectory to be partially faulty as is often the case with the corner matches in our experiments, neither the smoothness measure nor the fitting error measure is likely to give a good rating to the trajectory. Part of such trajectories are usable data for motion estimation. We developed a method where a trajectory is analyzed and spurious points are detected and deleted by clustering of points in the trajectory.

3.1 Refinement of Correspondence

Figure 2 illustrates the basic idea of refinement and linking of noisy correspondence data, where a box with a question mark represents a spurious point.

Trajectory 3 is highly noisy and thus useless. Parts of point trajectories 1, 4, 5 are faulty. The spurious points should be excluded from the input to motion estimation. The trajectory of a single feature is fragmented into shorter ones (trajectories 2 and 5) that should be linked into one. Otherwise they lose their identity and each one is treated as a separate feature. Furthermore, the total disparity vectors for fragmented features are

shorter than those from a complete sequence and thus the depth estimation is less stable.

The noisy data are continuously refined in the feedback loop. First, motion parameters are estimated using the raw correspondence data. Then, each noisy trajectory is partitioned into overlapping subsets of points each of which conforms to the estimated motion. The largest set is selected as the input for the next motion estimation. If a trajectory consists of good matches except for a few frames, then the bad points are detected. When a false match merged two separate (correct) trajectories into one, usually the larger correct set is found. If the trajectory is too noisy to produce any reliable subsets, then it is deleted. This selection process is repeated 3 or 4 times until a set of reliable correspondence is selected.

Refined correspondence are linked with the guidance from predicted motion. Linked data are refined again since there may be additional spurious points.

3.1.1 Selection of Good Points from a Trajectory

A trajectory is partitioned into sets of compatible points and the best set is selected. Partitioning is based on clustering of compatible point pairs rather than clustering of compatible points.

Given M, an estimate of motion, and a trajectory including points (A B C), a pair of points (A B) is defined to be able to *predict point* C if point C can be approximately obtained by extrapolating (A B) in accordance with M as described later. The tolerance in the approximate position of C depends on the 2-D distance of the point pair (A B) and the point C:

$$\|\hat{C} - C\| \leq \alpha \min(\|A - C\|, \|B - C\|),$$

where α is 0.25 in our experiments.

If a pair of points (A B) with a reliable motion estimate in a trajectory is a good match, then (A B) should predict the positions of other good points in the trajectory. In the example of figure 3, points (1 2 3 4 5 6) form a trajectory, where points (2 5) are the spurious points. Pairs (1 3) and (4 6) are compatible. and pairs (1 2) and (3 4) are not compatible.

Clustering is done by partitioning of a graph, where each node is a pair of points and the presence of an edge indicates that the two pairs are compatible.

The algorithm of clustering is given below, where a pair (A B) is defined to be *compatible to a cluster of pairs* if (A B) is able to predict all pairs of points in the cluster.

Clustering Algorithm

For each node in a graph
  if the node does not belong to a cluster
  then if there is a cluster with which
    the node is compatible
    then put the node into the cluster
    else generate a new cluster

After clustering, we get several fully connected subgraphs. Of the subgraphs with connectivity above a threshold, the set with the largest number of points is chosen for the input to the motion estimation in the next iteration. *Connectivity* is defined as:

$$\text{connectivity} = \frac{\text{number of edges}}{\text{number of points}}$$

Figure 4 shows the clusters from the trajectory in figure 3. All pairs of points (1 3 4 6) are compatible. The pairs (1 3) and (4 5) are compatible even though point 5 is a spurious point. Both subgraphs include the same number of points. The connectivity of the graph composed of points (1 3 4 6) is 3.75 and the connectivity of that including the spurious points is 0.25.

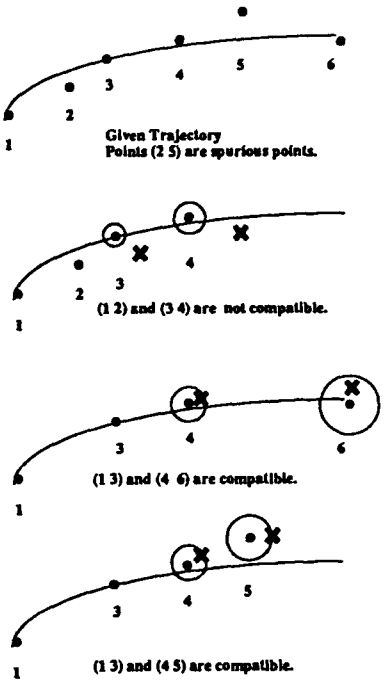


Figure 3: Compatibility between pairs of points in trajectory. X mark is the predicted location and the circles indicate allowable errors.

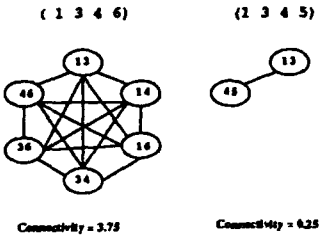


Figure 4: Selection of cluster

3.1.2 Extrapolation from a Point Pair

In order to extrapolate a point pair in the 3-D space, the depths for the points are necessary. Chronogeneous motion analysis used in this research produces the depth

of each point feature. This estimated depth is not used in the extrapolation since it is obtained for the entire (possibly noisy) trajectory. When the positions in the image plane are available for two frames, the depth of a point at each frame can be computed as follows:

Given 2 dimensional points P_i, P_j in the image plane and an estimate of motion parameters,

Compute the three dimensional points Q_i, Q_j which are projected onto P_i, P_j in the image plane.

We define

$T_{i,j}$: translation vector (from frame i to j)

$R_{i,j}$: rotation matrix (from frame i to j)

We let $Q_i = K_i P_i$ and $Q_j = K_j P_j$ as shown in figure 5. Then

$$R_{i,j} Q_i + T_{i,j} = Q_j$$

Hence

$$K_i R_{i,j} P_i + T_{i,j} = K_j P_j$$

Then

$$[\ R_{i,j} P_i | - P_j\ ] \begin{bmatrix} K_i \\ K_j \end{bmatrix} = -T_{i,j}$$

Let $A = [\ R_{i,j} P_i | - P_j\ ]$

Then

$$\begin{bmatrix} K_i \\ K_j \end{bmatrix} = -(A^t A)^{-1} A^t T_{i,j}$$

Finally, $Q_i = K_i P_i$ and $Q_j = K_j P_j$.

If either K_i or K_j is negative, the point pair (P_i, P_j) is not used in the clustering since it is on the wrong side of the image plane and in conflict with the estimated motion parameter.

The 3-D position of the point, Q_k, at an arbitrary frame k is computed either:

$$\hat{Q}_i^k = R_{i,k} Q_i + T_{i,k}$$

$$\hat{Q}_j^k = R_{j,k} Q_j + T_{j,k}$$

where $\hat{Q}_i^k$ is an estimate of Q_k with reference to Q_i.

Of the two estimates $\hat{P}_i^k$ and $\hat{P}_j^k$ which are the projections of $\hat{Q}_i^k$ and $\hat{Q}_j^k$ on the image plane, the better approximation of the point P_k is used in the compatibility computation.

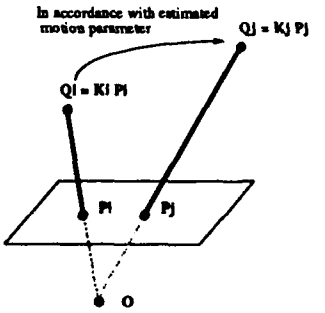


Figure 5: Qi,Qj are at the extension of Pi,Pj.

3.2 Linking of Correspondence

After the refinement of initial correspondence data, we get a set of refined point trajectories. Each point in a trajectory is marked as S (Selected) or D (Discarded). We define the *effective length* of a trajectory as the number of points marked S in it.

If linking is done only among already established trajectories, then the scope of possible matches is limited and an isolated corner does not have a chance of being linked with a trajectory. Hence we extend a trajectory whose effective length is longer than 3 and look for the best matching corner in each frame along the predicted extension. The predicted path for a point trajectory is obtained from the motion parameter estimation.

A window centered at the predicted position is set for each frame. Then, the match strengths of all the corners in the window are computed with respect to all the S corners in the trajectory. The match strengths are averaged and the best corner is selected as the one that is to be linked to the trajectory. Slight deviations of the properties of the S corners are smoothed in the averaging and thus the chance of wrong match decreases.

A simplified version of the matching function used in the initial corner match is used for the computation of the match strength. Only the internal properties of corners are used in the computation and the relations among corners are not considered. Local properties such as INSIDE-ANGLE, IN-DIRECTION and OUT-DIRECTION are given high weighting since the consistency of global properties is usually broken when a region match gets fragmented.

3.2.1 Properties of Linking

Let $\Pi = \{P_1, P_2, P_3, \ldots, P_m\}$ be the set of point trajectories. Under ideal conditions, Π is partitioned into $\Theta = \{T_1, T_2, T_3, \ldots, T_n\}$ where $T_i, 1 \leq i \leq n$, satisfies the following properties.

Property 1. Transitivity: If trajectory A is linked with trajectory B and B is linked with trajectory C, then A is linked with C. (A and C should not share common frames.)

Property 2. Symmetry: If the extension of trajectory A is linked with trajectory B, then the extension of B also is linked with A.

In most cases, the extension of disconnected corner matches meet both properties. However, when there are nearby objects with similar patterns, complicated linking happens which does not satisfy the properties. We do not use these links in motion estimation.

3.2.2 Refinement of Linked Matches

Linked correspondences consist of trajectories composed of S points and linked corners. Linked correspondences are refined in the same way as the initial correspondence to delete spurious points in the linking. Usually one refinement process is enough since the quality of linked data is much better than the initial data.

4 Results

The algorithm has been tested for standard sets of real image sequences. We present the result for the Rocket Field sequence provided by Dutta at UMASS [Dutta *et al.*, 1989]. Of the 30 frames of the sequence, we used the first 15 frames.

Typical region and corner matches are shown in figure 6. The dots mark the matched corner. In figure 6 (a), the object is a muddy area on terrain. The shapes for some of the frames are quite different from the rest and the positions of the corners are rather random. Still, part of the sequence of corner matching is usable, which forms the best cluster in the refinement process. In figure 6 (b), the corner is on the lower right of the building. Corner location error is hardly expected on this very clear boundary. However, the irregular motion of the vehicle caused the perturbation of the trajectory.

The initial and improved corner correspondence data used in motion estimation, superimposed on the image of the first frame of the sequence, are shown in figure 7. The start frame and end frame for those corner matches are arbitrary and so the end points of each trajectory may not point to any physical object on the underlying image. Figure 7 (a) represents the initial noisy correspondence. The black dot of a trajectory indicates the start position. The trajectories are noisy and some of them move in random direction. Its refinement is shown in figure 7 (b), where most of the noisy parts are deleted. The refined data are linked in figure 7 (c) and refined again as in figure 7 (d).

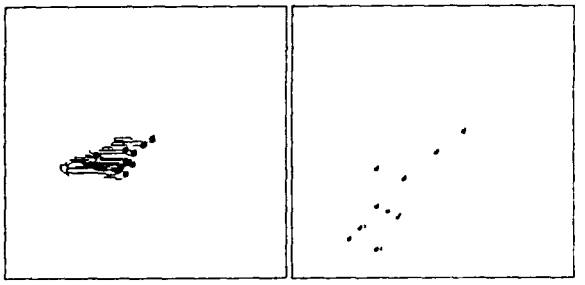
Figure 8 shows the reconstructed trajectories and the top view of the objects for which the ground truth is provided. The initial data are so noisy that some of the reconstructed trajectories are far from the real ones as shown in figure 8 (a), and the estimated depths of objects are unreliable as shown in figure 8 (b). The results from final linked and refined correspondence data are shown in figures 8 (c) and (d). The reconstructed trajectories are very close to the real ones and the reliability of the estimated depths are enhanced.

Table 1 is a list of the estimation error with reference to the ground truths when the final correspondence data are used. Most of them lie within 20 %. The estimated motion parameters for the final correspondence data are also shown in table 2.

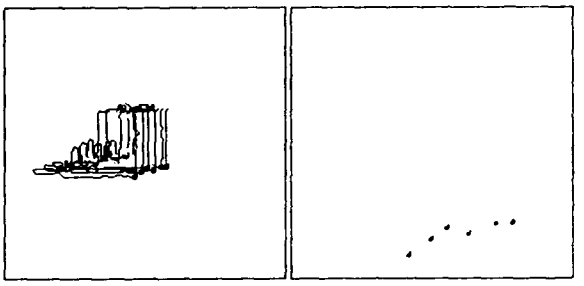
5 Conclusion

In this paper, we presented a feedback approach to improve the matching of features in the domain of feature-based motion analysis for multiple frames. We applied this approach in a motion analysis system that is built on hierarchical feature extraction and matching. We also described the sources and effects of errors in correspondence data found in the real image data. We showed that initial noisy correspondence data are gradually refined and fragments of a single feature are linked into one trajectory in a feedback loop where the feedback from 3-D motion estimation guides the prior stages.

We have applied this method to real image sequences of test data set and showed that the motion and structure estimation is improved due to the improved correspondence.



(a) Muddy area match



(b) Building match

Figure 6: Typical region and corner matches. The numbers near the dots in the right figures are the frame numbers.

References

[Adiv, 1985] G. Adiv. Determining 3 dimensional motion and structure from optical flow generated from several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(4), July 1985.

[Anandan, 1987] P. Anandan. A unified perspective on computational techniques for measurement of visual motion. In *Proceedings of the DARPA Image Understanding Workshop*, volume 2, pages 719–732, Los Angeles, California, 1987. Morgan Kaufmann Publishers, Inc.

[Broida *et al.*, 1990] T. J. Broida, S. Chandrashekhar, and R. Chellappa. Recursive 3d motion estimation from a monocular image sequecne. *IEEE Transactions on Aerespace and Electronic Systems*, pages 639–656, 1990.

[Chandrashekhar and Chellappa, 1991] S. Chandrashekhar and R. Chellappa. Passive navigation in a partially known environment. In *Proceedings of the Workshop on Visual Motion*, pages 2–7, 1991.

[Cheng and Aggarwal, 1990] C. Cheng and J. Aggarwal. A two-stage hybrid approach to the correspon-

Point	Depth	% Error
1	43.91	
	38.16	-13.08
14	49.32	
	48.92	-0.81
54	28.58	
	28.10	-1.69
91	35.50	
	43.16	21.56
130	27.11	
	24.84	-8.39
153	19.67	
	19.54	-0.70
157	56.42	
	68.78	21.91

Table 1: Reconstructed Structure Compared to Ground Truth Values for the Rocket Field Sequence

Parameter	x	y	z
Translation	3.7858	-3.3910	4.0264
Acceleration	-0.0003	0.0171	0.0130
Rotation (in degrees)	-0.0057	0.2841	0.2170
Rot-Center	-954.6281	-409.9108	511.4143

Table 2: Estimate of Motion Parameter for the Rocket Field Sequence at the final Frame

dence problem via forward-searching and backward-correcting. In *Proceedings of the International Conference on Pattern Recognition*, pages 173–177, 1990.

[Dutta *et al.*, 1989] R. Dutta, R. Manmatha, L. Williams, and E. Riseman. A data set for quantitative motion analysis. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 159–164, 1989.

[Faugeras and Price, 1981] O. D. Faugeras and K. Price. Symbolic description of aerial images using stochastic labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(6):633–642, November 1981.

[Fischler and Bolles, 1981] M. A. Fischler and R. C. Bolles. Random samples consensus: A paradigm for model filtering with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–396, June 1981.

[Fletcher *et al.*, 1991] M.J. Fletcher, K. Warwick, and R. J. Mitchel. The application of a hybrid tracking algorithm to motion analysis. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 84–89, 1991.

[Franzen, 1991] W. O. Franzen. *Structure from Chronogeneous Motion*. PhD thesis, University of Southern california, 1991.

[Jenkin, 1983] Michael Jenkin. Tracking three dimensional moving light displays. In *Proceedings of Workshop on Motion, Representation and Control. SIGART*, pages 66–70, 1983.

[Leung *et al.*, 1991] M. Leung, Y. Liu, and T. Huang. Estimating 3d vehicle motion in an outdoor scene from monocular and stereo image sequences. In *Proceedings of the Workshop on Visual Motion*, pages 62–68, 1991.

[Ohlander *et al.*, 1978] R. Ohlander, K. Price, and R. Reddy. Picture segmentation by a recursive region splitting method. *Computer Graphics and Image Processing*, 8:313–333, 1978.

[Roach and Aggarwal, 1980] J. W. Roach and J. K. Aggarwal. Determining the movement of objects from a sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(6):554–562, November 1980.

[Sethi and Jain, 1987] I. K. Sethi and R. Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(1):56–73, January 1987.

[Shariat and Price, 1990] H. Shariat and K. Price. Motion estimation with more than two frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(5):417–434, May 1990.

[Sull and Ahuja, 1991] S. Sull and N. Ahuja. Segmentation, matching and estimation of structure and motion of textured piecewise planar surfaces. In *Proceedings of the Workshop on Visual Motion*, pages 274–279, 1991.

[Tsai and Huang, 1984] R. Y. Tsai and T. S. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(1):13–26, January 1984.



(a) Initial correspondence



(b) Refined correspondence



(c) Linked correspondence

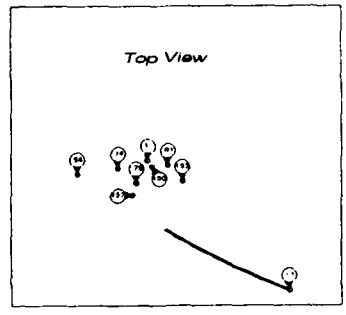


(d) Refined and linked correspondence

Figure 7: Improvement of correspondence data with each step



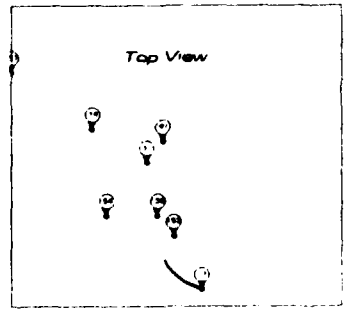
(a) Reconstructed from initial match



(b) Top view from initial match



(c) Reconstructed from final match



(d) Top view from final match

Figure 8: Reconstructed trajectory and top view of objects with ground truth. The curve numbered as -1 is the trajectory of the camera.

Structure from Chronogeneous Motion: A Summary *

Wolfgang Otto Franzen
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273

Abstract

We summarize the work presented in our dissertation. This includes contributions to the following areas of time varying image analysis: motion representation, structure from known motion (SFKM), structure from motion (SFM), and motion parameter estimation (MPE). Our research deals with multiframe monocular image sequences under perspective projection, and is feature based. We present a general methodology for developing efficient algorithms for the SFKM and SFM/MPE problems when motion is affine. The resulting algorithms treat points, lines, and occluded features in a uniform manner. Straightforward parallel versions of these algorithms may be implemented.

We introduce a class of motion that we call *chronogeneous* motion, and its associated matrix representation. Serial SFM/MPE algorithms have been implemented for several specific subclasses of affine motion. Among these is a very robust algorithm for the class of rigid chronogeneous motion.

1 Introduction

Motion analysis is an important area of research within the field of computer vision, that deals with sequences of two or more images. Some of the goals of motion analysis processing are to determine the coherently moving objects in the environment, to determine the shape of each object, and to determine the motion of the camera and/or each object. Our work deals specifically with the following motion analysis problems:

- *Motion Parameter Estimation (MPE) Problem*: Given a model of permissible motion, where perceived motion is modeled as being a projection of motion in 3-space, estimate the motion parameters for a camera or for a set of tokens that share a common motion.
- *Structure from Known Motion (SFKM) Problem*: Given a sequence of two or more images, and sets of corresponding tokens that share a common motion, *where the motion of these tokens is known*, determine the geometric relationships between tokens.
- *Structure from Motion (SFM) Problem*: Given a sequence of two or more images, and sets of corresponding tokens that share a common motion, determine the geometric relationships between tokens.
- *Depth Determination Problem*: Given a sequence of two or more images, and sets of corresponding tokens that share a common motion, determine the 3D position of each token relative to the viewer, for each frame.

The depth determination and SFM problems are closely related, and often authors do not distinguish between the two. We have actually solved the depth determination problem although we loosely claim to have solved the SFM problem.

The SFM and MPE problems are very important research topics in the area of time-varying image analysis. Efficient and robust solutions to these two problems have important robotics and automated vehicle applications. A solution to these problems allows the (scaled) structure and motion of moving objects to be recovered. Alternatively, the egomotion of a camera (and hence a vehicle) can be determined. Furthermore, a (sparse) environmental depth (or time to collision) map can be recovered. In addition, future feature image plane positions may be predicted, possibly aiding the correspondence process.

The SFKM problem may be viewed as a generalization of the stereo problem to multiple frames, where the interframe transformations are known. A solution to this problem is relevant to trinocular and slider stereo, and to attempts to determine object structure by moving a camera (or object) through a controlled sequence of poses. It is also relevant to passive ranging. In some cases camera motion can be determined quite accurately by means of inertial navigation sensors. When camera motion is accurately known, a solution to the SFKM problem allows better environmental depth estimates for stationary objects than when motion parameters and structure are estimated jointly.

The use of multiple (as opposed to two) frames is important for several reasons:

- the solution process tends to be more robust,

*This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-90-C-0078. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

- structure/motion can be recovered with fewer features being tracked, and
- higher order derivatives of motion can be estimated.

For this reason, the extensive work on two frame structure from motion, is not directly applicable. We want to develop algorithms that can recover motion when only a very small number of features are visible, and two frame linear methods (such as [Weng *et al.*, 1989]) require at least 8 correspondences, and preferably more.

2 Assumptions

We present solutions to the SFKM and SFM/MPE problems under a multiframe feature based paradigm. Our work deals specifically with monocular image sequences. We have solved these problems under the following constraints and/or assumptions:

- The input consists of a set of *trajectories*, where each trajectory gives the image plane position of a feature in each of several frames.
- Associated with each feature in each frame is an (externally supplied) inverse covariance matrix that determines the uncertainty in the image plane position of the corresponding feature. If this matrix is the zero matrix, then the feature is "occluded" in the corresponding frame. If this matrix is singular, then the feature is a line feature.
- A central projection (perspective) imaging model is assumed.
- The focal length and other camera calibration parameters are assumed to be known constants.[1]
- It is assumed that a stationary camera is imaging a moving object. However, it is shown how the results also apply for a camera moving through a static environment.
- All features are assumed to be undergoing the same motion.
- For our methodology to be applicable, 3D feature position must be transformed in an affine manner. However, the assumed class of motion may be any *subclass of affine motion*.

Specifically, object structure is *not* restricted in any way.

3 The Methodology

Our general approach is as follows. The SFKM and SFM/MPE problems are formulated as function minimization problems. Through proper selection of the function to be minimized, it is possible to determine the structural unknowns in closed form in terms of the motion parameters. This results in a large reduction in the number of unknowns, and leads to a class of algorithms with very low time complexity. Furthermore, the computations for different features are similar, largely independent of one another, and interact mainly through the motion parameters. Therefore, potentially large speedups can be achieved through parallel processing.

We now discuss our methodology in more detail. First, let us introduce some notation.[2] Let

n_t be the total number of features being tracked (indexed 0 through $n_t - 1$),

n_f be the number of frames (indexed $-(n_f - 1)$ through 0),[3]

$\vec{Q}^s_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j})^t$ be the spatial (3D) position of the j^{th} feature in the i^{th} frame,

$\vec{Q}^p_{i,j} = (x_{i,j}/z_{i,j}, y_{i,j}/z_{i,j})^t$ be the image plane coordinates corresponding to $\vec{Q}^s_{i,j}$,

$\vec{Q}^m_{i,j} = (X^m_{i,j}, Y^m_{i,j})^t$ be the actually measured image plane coordinates of the j^{th} feature in the i^{th} frame,

$\Sigma_{i,j}$ be the (symmetric) (2×2) covariance matrix associated with the image plane location uncertainty of the j^{th} feature in the i^{th} frame, and let $\Sigma_{i,j}{}^{-1}$ be its inverse, and let

$\mathcal{F}_{i,j}$ be the symmetric (3×3) matrix $\begin{bmatrix} 1 & 0 & -X^m_{i,j} \\ 0 & 1 & -Y^m_{i,j} \end{bmatrix}^t \Sigma_{i,j}{}^{-1} \begin{bmatrix} 1 & 0 & -X^m_{i,j} \\ 0 & 1 & -Y^m_{i,j} \end{bmatrix}$.

Using the preceding notation, the image plane error norm is defined as follows:

$$f_{opt} = \tfrac{1}{2} \sum_{i=-(n_f-1)}^{0} \sum_{j=0}^{n_t-1} (\vec{Q}^p_{i,j} - \vec{Q}^m_{i,j})^t \Sigma_{i,j}{}^{-1} (\vec{Q}^p_{i,j} - \vec{Q}^m_{i,j}) \quad (1)$$

This norm may also be written as

$$f_{opt} = \tfrac{1}{2} \sum_{i=-(n_f-1)}^{0} \sum_{j=0}^{n_t-1} \frac{\vec{Q}^s_{i,j}{}^t \mathcal{F}_{i,j} \vec{Q}^s_{i,j}}{z^2_{i,j}} \quad (2)$$

Ideally, the image plane error norm is the function that should be minimized. However, this leads to nonlinear optimization problems in large numbers of unknowns, with certain attendant difficulties. Ando [Ando, 1991] has recently pointed out that this norm is quadratic in the x and y components of 3D position, and therefore these components can be computed in closed form in terms of the motion parameters and unknown z-components. This still leaves considerable nonlinearity, a potentially large (although significantly reduced) number of unknowns, and the problem of providing good initial guesses for all the parameters.

We have tried an alternate approach. The image plane error norm is replaced by one of the following two

[1] In the SFM/MPE case, the focal length need not be known accurately, as in the absence of certain additional information, the absolute scale cannot be determined anyway. A rough estimate of the focal length is satisfactory, provided the focal length is constant. In addition, our methodology could easily be extended to the case where the focal length varies, provided the ratios of the focal lengths for the various frames were known accurately.

[2] In the following, for simplicity, the focal length is taken to be be unity.

[3] Under this indexing scheme, the *final* frame of the image sequence has an index of 0, and all preceding frames have a negative index.

approximations.[4] We refer to the first of these approximations as the *pseudo-perspective* norm[5]:

$$f_{pseudo} = \frac{1}{2} \sum_{i=-(n_f-1)}^{0} \sum_{j=0}^{n_t-1} \frac{\vec{Q}^s_{i,j}{}^t \mathcal{F}_{i,j} \vec{Q}^s_{i,j}}{z^2_{0,j}}$$

$$= \frac{1}{2} \sum_{j=0}^{n_t-1} \frac{1}{z^2_{0,j}} \sum_{i=-(n_f-1)}^{0} \vec{Q}^s_{i,j}{}^t \mathcal{F}_{i,j} \vec{Q}^s_{i,j} \quad (3)$$

We refer to the second as the *quadratic* norm:

$$f_{quad} = \frac{1}{2} \sum_{i=-(n_f-1)}^{0} \sum_{j=0}^{n_t-1} \vec{Q}^s_{i,j}{}^t \mathcal{F}_{i,j} \vec{Q}^s_{i,j} \quad (4)$$

Both these norms have the property that they are quadratic functionals of the structural unknowns. For the pseudo-perspective norm, the structural unknowns are the estimates of the refined image plane location and inverse depth (inverse z-component) of each feature in the zeroth frame. For the quadratic norm, the structural unknowns are merely the estimates of the 3D position of each feature in the zeroth frame. In addition, there are certain linear constraints, that we do not discuss further at this point (see [Franzen, 1991b]).

Therefore, for either of these norms, the SFKM problem may be solved in closed form, as the problem to be solved is a quadratic programming problem subject to linear constraints.[6] Using this result, it is possible to develop iterative algorithms to solve the SFM/MPE problem for any given subclass of affine motion, with only the motion parameters as unknowns. This decoupling of the estimation of the motion parameters and the structure results in a large reduction in the number of unknowns. Cui, Weng, and Cohen [Cui *et al.*, 1990], who approach the problem differently, achieve a similar decoupling. Although, in general, the resulting SFM/MPE problem is nonlinear in the motion parameters, a closed form solution is possible for certain specific classes of motion.

Algorithms developed using this methodology have a very low time complexity. The time to solve the SFKM problem is proportional to the number of correspondences. For cases where the SFM/MPE problem can be solved in closed form, the computational complexity is of the same order. For iterative solutions to the SFM/MPE problem, the time for each iteration is proportional to the number of correspondences. When the number of motion parameters is fixed, independent of the number of frames, the number of iterations is bounded in practice, and therefore the total time is again proportional to the number of correspondences.

[4] In [Franzen, 1991b, section 5.1.3], two *pairs* of approximations are introduced, where the members of each pair are intimately related.

[5] One may choose any particular frame as the one from which the replicated (inverse) z weights are used. In practice, we choose the middle frame of the image sequence rather than the zeroth.

[6] The motion must be affine in order for this result to go through.

4 Are Solutions Significantly Biased?

A valid concern is that significantly biased solutions might occur as a result of minimizing these alternate norms. Based on real data and extensive simulation results, the following claims/observations may be made.

First, if the quadratic error norm is minimized directly, the resulting solutions are, indeed, generally unsatisfactory and may have significant bias. This was the case for motion in the class of uniform 3D acceleration. In order to improve the quality of the solutions, a principled bias correction scheme was devised for both the quadratic and the pseudo-perspective norms (see [Franzen, 1991b, sections 4.4 and 5.1.2] for details). For motion in the class of uniform relative 3D acceleration, the resulting improvement was quite substantial. The bias correction consists of adding some terms that are quadratic functions of the structural unknowns. In the presence of noise, these added terms generally cause the minima of these alternate norms to more closely coincide with those of the image plane error norm.

Second, the bias-corrected pseudo-perspective norm performs optimally and no significant improvement could be achieved by minimizing the image plane error norm instead. This claim is based on a statistical analysis of data from synthetic test cases for chronogeneous motion (see below). Actual residuals were in excellent agreement with theoretically predicted residuals.

Third, some of our undocumented synthetic experiments indicate that when the image plane error norm is minimized, there is a tendency to get trapped in local minima. Although anecdotal, our experience seems to indicate that energy surfaces for this norm tend to be somewhat bumpy near global minima. This problem does not seem to be shared by the alternate norms. Because these alternate error norms are quadratic functionals in the structural unknowns, the resulting error surfaces seem to be more well-behaved.

5 Chronogeneous Motion

Chronogeneous motion was introduced in [Franzen, 1988; Franzen, 1989; Franzen, 1991b]. This class of motion has 15 degrees of freedom in the general case, and 7 degrees of freedom in the rigid case. Rigid chronogeneous motion includes uniform acceleration and constant angular velocity rotation and translation as special cases.

Chronogeneous motion has the following important properties:

- *affine transformation of coordinate space* – this permits application of our more general results to derive efficient algorithms for the SFKM and SFM/MPE problems for this class of motion.
- *matrix representation* – this allows computations to be expressed in a succinct manner, namely using a (5×5) transformation matrix.
- *unique representation* – corresponding to each chronogeneous motion there is exactly one chronogeneous transformation matrix and *visa versa*.
- *fixed number of motion parameters independent of the number of frames* – this enhances the stability of the recovered solutions.

- *interesting class of motion* – chronogeneous motion is sufficiently general to model commonly occurring types of motion, yet not so general as to require an inordinate number of parameters to be determined.

As homogeneous coordinates are transformed by homogeneous matrices, so also are *chronogeneous coordinates* transformed by chronogeneous matrices. Time is the additional (fourth) component of these 5 dimensional vectors.

In [Franzen, 1988; Franzen, 1989; Franzen, 1991b], the properties of the chronogeneous matrix representation are investigated at some length. For instance, it is shown how to straightforwardly compute motion parameters such as axis and rate of rotation, center of rotation, velocity, and acceleration magnitude[7] given a (rigid) chronogeneous transformation matrix, and *visa versa*.

6 Particular Implementations

We have devised solutions to the SFM/MPE problem for several different subclasses of affine motion, namely uniform rotation about the optical center of the camera, uniform relative 3D acceleration, and rigid chronogeneous motion. A computer program was written that implements these methods in their full generality, except that (true) line features are not currently supported.

6.1 Chronogeneous motion

We have developed an iterative algorithm to recover structure and motion parameters, when the motion is rigid chronogeneous. The rigid chronogeneous SFM algorithm is extremely robust, and almost always finds the "correct" solution, or one with a residual that is just as good. This is in large part due to our scheme for generating multiple (eight) initial guesses, which almost always results in the global minimum of the objective function being found.

Extensive simulations were performed on the rigid chronogeneous SFM algorithm, as well as some tests on real data. Synthetic tests investigated algorithm performance as a function of number of frames, structure and type of object being imaged, object size, and to a lesser extent resolution and noise. We can only give a very brief summary of these results here. For the full test results see [Franzen, 1991b, Chapter 6].

6.1.1 Synthetic test results

For the experiments in which the number of frames was varied, the actual extent of motion was held constant in each test case, but sampling occurred at a correspondingly higher rate when the number of frames was increased. There was a very significant improvement in the accuracy of solutions when the number of frames was increased from 3 to 7, a small but significant improvement from 7 to 11 frames, and very little improvement from 11 to 15 frames.

Certain experiments showed that when the imaged structure was planar there were alternate valid interpretations. This result was not unexpected. Variations in environmental depth of roughly 10% to 15% were required to eliminate the ambiguity. In general, it was found that in order to obtain good and unambiguous reconstructions, a large field of view and adequate deviation from planarity were much more important than imaging large numbers of points *per se*. In addition, even for certain nonplanar configurations, if the motion in depth was less than 10% to 15% over the sequence, there was some possibility for ambiguity.

It was found that for a cube, a wicker reversed interpretation always exists as a local minimum of the objective function. For a cube subtending a small angle, or whose motion in depth is small compared to the dimensions of an edge, this alternate interpretation occasionally has a lower residual than the "correct" interpretation due to noise.

In general, in terms of being able to recover egomotion and being able to estimate times to collision, the algorithm performs quite satisfactorily (except near the focus of expansion, or for extremely distant features) if errors on the order of 5% to 10% are deemed acceptable. For this task, an image resolution of 512 by 512 is marginal. Correspondingly better results could be achieved at a 1K by 1K, or 2K by 2K resolution.

6.1.2 A real test case

The Rolling Tire sequence was first analyzed by Broida in section 8.3 of his dissertation [Broida, 1987], where he referred to it as the Car sequence. Eight features (adhesive dots) were placed on the right front tire of a car. The car was translated (roughly) uniformly from left to right, while also approaching the camera. Therefore, the features were undergoing a cycloidal type motion in the image plane. Figure 1 shows the first and last frames of this image sequence.

The rigid chronogeneous SFM algorithm finds three different clusters of solutions. The worst of these solutions has an optimal residual well over sixty thousand times the best, and is a totally spurious solution. The second best solution still has a residual more than fifteen times the best, and is not a serious alternative to the best solution.

Tables 1 and 2 display respectively the computed external motion parameters and computed structure for the best solution. For this solution, a small *negative* vertical velocity component is found, which is consistent with the fact that the camera was pointed slightly downward. Figure 2 shows the *approximate* reconstructed image plane trajectories[8] for the best solution found by the rigid chronogeneous SFM algorithm, when overlaid on the *initial* frame of the image sequence.

We now compare our results to those computed by Broida's batch algorithm. Table 3 compares the interpoint distances computed by our algorithm to those com-

```
External motion parameters for final frame:
spin (deg/frame): ( -9.857159,  2.648302, 10.779173)
   center of rot: (  1.288754, -0.750319,  1.362862)
        velocity: ( -0.192815, -0.042104, -0.087495)
    acceleration: ( -0.001222,  0.000328,  0.001337)
signed acceleration magnitude:  0.001841
```

Table 1: Computed External Motion Parameters for the Rolling Tire Sequence

[7] We refer to these as the *external* motion parameters.

[8] The input data was preprocessed and there was some difficulty getting the scaling and the offset correct.

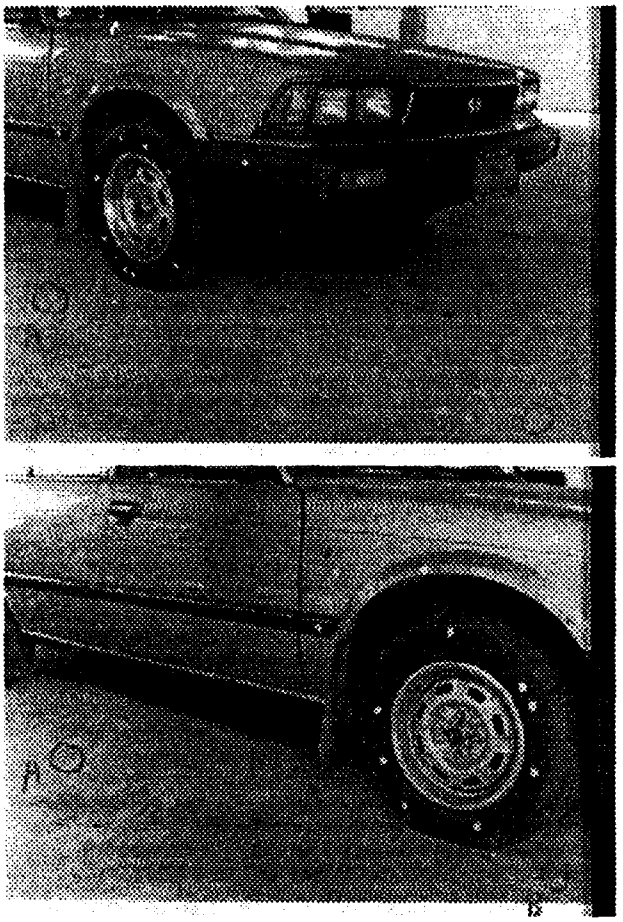


Figure 1: First and Last Frames (Frames 0 and 15) of Rolling Tire Sequence

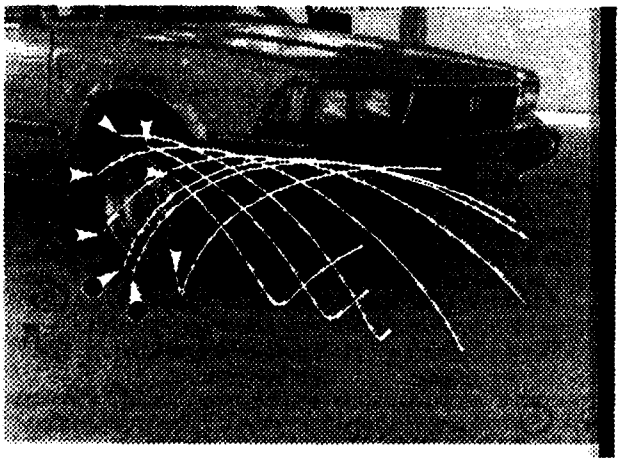


Figure 2: Reconstructed Trajectories for the Rolling Tire Sequence

```
Structure for initial frame:
POINT   0: (  1.5251, -0.0786,  5.8790)
POINT   1: (  1.6455, -0.0243,  5.9683)
POINT   2: (  1.9061,  0.3491,  6.1166)
POINT   3: (  1.9349,  1.0020,  6.0065)
POINT   4: (  1.5632,  1.3655,  5.6021)
POINT   5: (  1.2515,  1.2271,  5.3752)
POINT   6: (  0.9885,  0.9295,  5.2138)
POINT   7: (  0.9841, -0.0353,  5.4382)

Structure for final frame:
POINT   0: ( -1.7093,  0.3168,  3.7162)
POINT   1: ( -1.7757,  0.1757,  3.6827)
POINT   2: ( -1.8029, -0.2935,  3.7747)
POINT   3: ( -1.5095, -0.7328,  4.1750)
POINT   4: ( -1.0292, -0.6294,  4.6137)
POINT   5: ( -0.8608, -0.2668,  4.7026)
POINT   6: ( -0.7994,  0.1555,  4.6610)
POINT   7: ( -1.2910,  0.7641,  4.0534)
```

Table 2: Computed Structure for the Rolling Tire Sequence

puted by Broida's algorithm, and to the ground truth.[9] Our solution has been rescaled so that the average computed interpoint distance is equal to the average interpoint distance for the ground truth. Our results are *much* better than Broida's. Our largest interpoint distance error is less than 2.4%. Many of Broida's distances have errors of roughly 50%, and the largest error is nearly 74%.

Since the tires were (roughly) pointed in the direction of travel, the angle between the axis of rotation and the velocity vector should be 90 degrees. The angle computed by our algorithm was 74.7 degrees. Broida's algorithm computed an angle of roughly 150 degrees. The total rotational angle was measured to be 3.85 radians. This compares to our result of 3.89 radians, and Broida's result of 3.96 radians. The total translational distance was measured to be 45.0 inches. Broida's result was 43.5 inches, and ours was 39.0 inches, which was reasonable considering the CRLB estimated standard deviation was 5.5 inches.

The only plausible explanation for our algorithm performing so much better than Broida's is that his algorithm converged to a local minimum in this particular case. However, the solution found by his algorithm does not correspond to any of the solutions found by our algorithm. Perhaps the image plane norm (which Broida's algorithm was minimizing) and the pseudo-perspective norm have different nonglobal minima, or perhaps our algorithm did not find that particular local minimum.

6.2 Uniform 3D acceleration

When motion lies in the class of uniform 3D acceleration, a closed form algorithm results when the quadratic norm is minimized. The algorithm involves finding the eigenvector corresponding to the minimum eigenvalue of

[9] The ground truth and the estimates generated by Broida's algorithm were extracted from Table 8.5 on page 183 of his dissertation [Broida, 1987]. We have corrected a couple of typographical errors that appeared in the original (private communication with the author).

i	j	Ground	Broida	% Err	Franzen	% Err
0	1	1.94	1.09	-43.81	1.95	0.53
0	2	7.63	7.20	-5.64	7.58	-0.61
0	3	14.19	20.66	45.60	14.22	0.20
0	4	18.13	28.38	56.54	17.98	-0.80
0	5	17.50	27.40	56.57	17.44	-0.36
0	6	16.13	23.42	45.20	16.16	0.18
0	7	8.44	5.21	-38.27	8.55	1.28
1	2	5.94	6.76	13.80	5.86	-1.42
1	3	13.06	20.54	57.27	13.05	-0.10
1	4	17.81	28.42	59.57	17.60	-1.17
1	5	17.75	27.52	55.04	17.61	-0.82
1	6	16.94	23.66	39.67	16.90	-0.23
1	7	10.25	6.17	-39.80	10.37	1.12
2	3	8.13	14.13	73.80	8.10	-0.31
2	4	14.88	22.36	50.27	14.55	-2.24
2	5	16.44	21.76	32.36	16.17	-1.63
2	6	17.38	18.40	5.87	17.27	-0.65
2	7	14.69	8.81	-40.03	14.77	0.51
3	4	8.25	8.57	3.88	8.05	-2.38
3	5	11.75	8.72	-25.79	11.70	-0.39
3	6	15.06	7.72	-48.74	15.12	0.41
3	7	18.25	19.44	6.52	18.56	1.68
4	5	5.00	2.44	-51.20	5.01	0.16
4	6	9.88	6.71	-32.09	10.02	1.39
4	7	18.38	26.23	42.71	18.64	1.42
5	6	5.19	4.65	-10.40	5.24	1.02
5	7	15.50	24.88	60.52	15.80	1.92
6	7	11.88	20.53	72.81	12.11	1.95

Table 3: Interpoint Distances for the Rolling Tire Sequence

a certain (6×6) matrix. The motion parameters and structure are recovered by applying appropriate linear transformations to this eigenvector. The six eigenvectors found by the algorithm are used to generate six of the initial guesses for the rigid chronogeneous SFM algorithm.

This algorithm is described in detail in [Franzen, 1991b, Chapter 4] and in [Franzen, 1991a]. This algorithm is quite fast and could run in real time on current generation hardware, if a processor per feature were available.

6.3 Pure rotation

We have developed an iterative algorithm that recovers refined image plane position and rotational motion parameters when the motion consists of a uniform rotation about the optical center of the camera. This algorithm minimizes the pseudo-perspective norm. It is used to generate one of the initial guesses for the rigid chronogeneous SFM algorithm.

The pure rotation algorithm converges quite fast, but there is room for further improvement, as it does not always find the global minimum. By imposing certain restrictions, a closed form quaternion method could be used to generate a better initial guess than the all zero guess that we have used. See [Franzen, 1991b, sections 5.3.3 and 7.3.6] for further details.

7 "Danger – Will Robinson"[10]

It is the nature of published research that what manifestly does not work generally doesn't get published. I would like to take his opportunity, however, to indicate some things that caused myself a great deal of wasted effort, and perhaps are not generally appreciated, so that other researchers may benefit.

At a certain point in our research, we tried to develop an iterative SFM/MPE algorithm that could operate when arbitrary interframe rigid transformations were allowed. The algorithm attempted to minimize the image plane error norm. This algorithm had tremendous difficulty locating the global minimum when unintelligent initial guesses were used. In addition, in synthetic experiments when known original (noise free) configurations were used as initial guesses, during the process of minimization the solutions tended to diverge from what was expected (although the residual was indeed reduced). Admittedly, small numbers of points (say 8) were being used. However, this has a bearing on recursive methods such as [Cui *et al.*, 1990] that compute a different rigid transformation for each frame. Such methods *may* be unstable/unreliable unless many features are tracked.

When the rigid chronogeneous SFM algorithm was first being tested, there was only one initial guess, namely the solution found by the closed form uniform 3D acceleration algorithm. In simulations, sometimes the original chronogeneous algorithm gave good results, and sometimes poor results. Since theoretical expected values of the residuals had been derived, and the computed residuals were on average too high (and tended to be distributed in clusters), something was clearly amiss. We determined that the algorithm was finding alternate local minima. For certain test configurations, the algorithm would converge to one of several different minima as the added image plane noise was altered from one test case to the next.[11] We hoped that by devising a more sophisticated initial guessing scheme, or by adding appropriate biasing terms, or by tweaking some of the internal parameters of the numerical optimization routine this problem could be eliminated. Way too much time was spent on this endeavor. Although certain modifications improved the results, try as we might, we couldn't hit upon any scheme of generating *single* initial guesses that converged to the global minimum over a wide range of motions and structures more than roughly 90% of the time. It was only after an intelligent scheme for generating multiple initial guesses was devised that robust behavior was achieved.

Many iterative computer vision algorithms have been developed that minimize nonlinear functions. We don't think that researchers have adequately addressed whether, with what frequency, and under what conditions global minima are actually found. Unless a very good initial guess is available by some other mechanism, or one has taken particular care to devise a good system of initial guesses, one should realize that nonglobal

[10] Some of you may remember the television series *Lost in Space*. Among other things, it featured a robot that often issued dire warnings – unfortunately usually too late to be of any real assistance.

[11] Commonly, two or three clusters of solutions were found. For one early test configuration, there seemed to be four clusters.

minima may be found a nonnegligible fraction of the time. When they are applicable, it is for this reason that (largely) closed form algorithms such as [Spetsakis, 1991] (which handles both point and line features over exactly 3 frames) and [Tomasi and Kanade, 1991] (which recovers arbitrary rigid transformations and shape over multiple frames, but assumes orthographic projection) have great utility.

8 Summary

We have introduced a methodology for solving the SFKM and SFM/MPE problems. The approach is one of function minimization, where the function to be minimized is either the *pseudo-perspective* norm or the *quadratic* norm. The resulting algorithms have the following desirable properties:

- An arbitrary number of features and frames are permitted (provided only that information sufficient to properly constrain solutions must be supplied).
- Both point and line features are handled within the same framework.
- A given feature need not be visible in every frame and so the method works in the presence of occlusion or correspondence "drop outs."
- It is possible to model the uncertainty in feature position due to motion blur.
- Serial computation times are generally proportional to the number of correspondences.
- Efficient parallel implementations can be developed in a straightforward manner.

Researchers should consider minimizing either of the aforementioned norms in other contexts, especially the (bias-corrected) pseudo-perspective norm. These norms seem to have smoother and more well-behaved energy surfaces than the image plane error norm. The quadratic norm should not be minimized without bias correction.

Chronogeneous motion, chronogeneous transformation matrices, and also chronogeneous coordinates were introduced. The chronogeneous matrix representation has many interesting properties which are discussed further in [Franzen, 1988], [Franzen, 1989], and [Franzen, 1991b, Chapter 3].[12]

SFM/MPE algorithms were implemented for the following classes of motion: uniform rotation about the optical center of the camera, uniform 3D acceleration, and rigid chronogeneous motion. The rigid chronogeneous SFM algorithm is very robust.

We note that several solution error metrics not otherwise discussed here were introduced in our dissertation. These were arrived at after much consideration, and dissatisfaction with commonly used error metrics such as mean squared 3D error. Researchers may find them useful in evaluating the performance of algorithms. See [Franzen, 1991b, section 6.1] for details.

References

[Ando, 1991] Hiroshi Ando. Dynamic reconstruction of 3D structure and 3D motion. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 101–110. IEEE Computer Society Press, Oct 7–9, 1991.

[Broida and Chellappa, 1991] Ted J. Broida and Rama Chellappa. Estimating the kinematics and structure of a rigid object from a sequence of monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13(6):497–513, June 1991.

[Broida, 1987] Theodore James Broida. *Estimating the Kinematics and Structure of a Moving Object from a Sequence of Images.* PhD thesis, Department of Electrical Engineering, University of Southern California, December 1987. Available as USC-SIPI Report #117.

[Cui *et al.*, 1990] Ning Cui, Juyang Weng, and Paul Cohen. Extended structure and motion analysis from monocular image sequences. In *Proceedings of the Third International Conference on Computer Vision*, pages 222–229. IEEE Computer Society Press, December 4–7, 1990.

[Franzen, 1988] Wolfgang O. Franzen. Natural representation of motion in space-time. In *Proceedings of the Image Understanding Workshop*, pages 443–452. Morgan Kaufmann Publishers, Inc., April 6–8, 1988.

[Franzen, 1989] Wolfgang O. Franzen. Representing motion using chronogeneous transformations. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 282–289. IEEE Computer Society Press, March 20–22, 1989.

[Franzen, 1991a] Wolfgang Otto Franzen. Structure and motion from uniform 3D acceleration. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 14–20. IEEE Computer Society Press, Oct 7–9, 1991.

[Franzen, 1991b] Wolfgang Otto Franzen. *Structure from Chronogeneous Motion.* PhD thesis, Department of Computer Science, University of Southern California, May 1991. Available as USC-IRIS Report #266.

[Shariat and Price, 1990] Hormoz Shariat and Keith E. Price. Motion estimation with more than two frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):417–434, May 1990.

[Shariat, 1986] Hormoz Shariat. *The Motion Problem: How to use more than two frames.* PhD thesis, Department of Electrical Engineering, University of Southern California, October 1986. Available as USC-IRIS Report #202.

[Spetsakis, 1991] Minas Spetsakis. Getting around the aperture problem using point and line correspondences. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 236–241. IEEE Computer Society Press, Oct 7–9, 1991.

[Tomasi and Kanade, 1991] Carlo Tomasi and Takeo Kanade. Factoring image sequences into shape and motion. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 21–28. IEEE Computer Society Press, Oct 7–9, 1991.

[Weng *et al.*, 1989] Juyang Weng, Thomas S. Huang, and Narendra Ahuja. Motion and structure from two perspective views: Algorithms, error analysis, and error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(5):451–476, May 1989.

[12] There have been minor changes in notion and sign conventions over the course of time, and the version in [Franzen, 1991b] should be taken as definitive.

Motion Segmentation Using Singular Value Decomposition

Terrance E. Boult and Lisa Gottesfeld Brown
Center for Research in Intelligent Systems
Dept. of Computer Science, Columbia University, NYC NY 10027
boult@cs.columbia.edu brown@division.cs.columbia.edu

Abstract

This paper addresses the problem of motion segmentation using the Singular Value Decomposition of a feature track matrix. It is shown that, under general assumptions, the number of numerically nonzero singular values can be used to determine the number of motions. Furthermore, motions can be separated using the right singular vectors associated with the nonzero singular values. A relationship is derived between a good segmentation, the number of nonzero singular values in the input and the sum of the number of nonzero singular values in the segments. The approach is demonstrated on real and synthetic examples. The paper ends with a critical analysis of the approach.

1 Introduction and Previous Work

The use of "motion" information has been around for many years, and considerable research progress has been made. Little of this work on motion, however, addresses the issue of segmenting the motions in a scene into different components but rather assumes, often implicitly, that that the images have been previously segmented into their constituent motions. Previous work on motion segmentation includes:

- intensity based image segmentation using known depth [Peleg and Rom, 1990] [Yamamoto, 1990],
- looking for "boundaries" in "optic-flow fields" (i.e. assuming locally uniform motion, but not knowledge of depth), [Shizawa and Mase, 1990] [Adiv, 1985], [Murray and Buxton, 1987],
- clustering in some type of *a priori* defined parametric motion space [Fennema and Thompson, 1979], [Dickmanns, 1989],
- techniques using Markov Random Fields to find both a segmentation and description of a motion sequence [Heita and Bouthemy, 1990], [Subrahmonia *et al.*, 1990],
- and a technique which looks for 2 motion components in an image sequence [Bergen *et al.*, 1990b], [Bergen *et al.*, 1990a].

Much of this work has made restrictive assumptions on the scene/motion to allow segmentation or assumes considerable *a priori* information (e.g. a depth map).

The approach we introduce in this paper, segments the motions in a scene into their different rigid body motions without knowledge of camera or object motion, shape, or depth. We assume an input of tracked features and, except for the smoothness necessary for that tracking, the approach to segmentation and motion/shape recovery does not require smoothness assumptions on either the objects or the motion. To obtain these desirable features we use the elegant "factorization approach" to shape/motion recovery. The idea of using SVD factorization of motion tracks was recently introduced (see [Tomasi and Kanade, 1990a],[Tomasi and Kanade, 1990b]) and is discussed in more detail elsewhere in these proceedings.

The paper is organized as follows: we first discuss the SVD in a little more detail. We then present a short review of Tomasi and Kanade's ground breaking work on SVD factorization and motion recovery. This material in hand, we describe our approach in Section 3. Section 4 presents some experimental testing of the approach, followed by a critical analysis in Section 5. We end with some conclusions and a discussion of future work.

2 Backgroun[illegible]

In this sectic[illegible]esent background material on the singula[illegible]decomposition and on the motion factoriza[illegible]echnique.

2.1 Some properties of the SVD

Assume, without loss of generality, I is an $m \times p$ matrix, $m \geq p$. By computing the Singular Value Decomposition (SVD) of the input I we can obtain 3 matrices L, Σ, R such that $I = L\Sigma R^\top$. Furthermore, $L = [l_1, l_2, \ldots l_p]$ is an $p \times p$ orthogonal matrix, $R = [r_1, r_2, \ldots r_p]$ is an $n \times p$ orthogonal matrix, and Σ is a diagonal $p \times p$ matrix $diag(\sigma_1, \sigma_2, \ldots \sigma_p)$ with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$. The value σ_i is referred to as the ith singular value, and the column vectors l_i, r_i are, respectively, the ith left singular vector and the ith right singular vector.

We now state some properties of the SVD which will be useful in the remainder of the paper. There is a strong relationship between SVD and eigenvalues / eigenvectors. In particular, the singular values are the eigenvalues of $I^\top I$ and the right singular vectors are its eigenvectors. Since $I^\top I$ is a symmetric matrix, the singular values are also the squares of the eigenvalues of I. To help in visualizing how the SVD extracts much of the structure inherent in a matrix, it is also helpful to know that the singular values of the matrix I are precisely the lengths of the semi-axes of the hyperellipsoid E defined by $E = \{y | y = Ix, \|x\|_2 = 1\}$. Furthermore, the location of the axes of this ellipsoid are given by the columns of L. Thus the SVD is strongly related to the principal component analysis of the data in I.

The relationship between SVD and principal components explains why we are able to perform motion segmentation using clustering of the right singular vectors. Principal component analysis is generally performed to reduce the dimensionality of a data set with many interrelated variables to a much smaller set, the principal components, while retaining as much of the original variation as possible. Each principal component is composed of a linear function α which operates on the vector x of random variables and maximizes the variance (and is uncorrelated with previously found components). The kth principal component is given by $\alpha_k^T x$ where α_k is an eigenvector of the covariance matrix of x corresponding to its kth eigenvalue. Since the right singular vectors of the SVD are the eigenvectors of $I^\top I$, they are precisely the principal components. In our case, each random variable is the location of a feature point given at different times. Random variables are related in a definite way according to whether they are part of the same motion. This is ultimately expressed in the principal components.

If the SVD of I is such that $\sigma_1 \geq \ldots \geq \sigma_k > \sigma_{k+1} = \ldots \sigma_p = 0$, then we know, $Rank(I) = k$ and $I = \sum_{i=1}^{k} \sigma_i l_i \otimes r_i^\top = \hat{L}_k \hat{\Sigma}_k \hat{R}_k^\top$. When the matrix I is noisy, the issue of determining "numerical" rank, call it $\mathcal{R}(I)$, is more difficult and will be discussed in section 3.5.

Finally, we comment on the computational complexity of the SVD algorithm. For our needs, the straightforward approach to the SVD costs around $7mp^2 + 4p^3 + O(p^2 + mp)$ FLOPS for a matrix with m rows and p columns. Code for computing the SVD can be found in any good numerical package (LINPACK, EISPACK, NAG, IMSL). We use a locally modified version of the code from Numerical Recipes in C [Press *et al.*, 1988].

2.2 Background on the shape/motion factorization technique.

Our discussion of the factorization technique has been broken into four terse subsections. We will treat both the 2D case (motion restricted to a plane) and the 3D case. The rationale for this is that the 2D case is significantly easier to visualize and present while the methods are very similar in practice.

All of the conceptual content of this section follows from the work of Tomasi and Kanade. For the 2D problem we follow [Tomasi and Kanade, 1990c] and for the 3D problem we follow [Tomasi and Kanade, 1991]. This section is intended to introduce notation for our development, and does not thoroughly explore the factorization-based approach.

The following assumptions are needed to make the approach feasible:

- the imaging system is orthographic,
- there are at least 3 frames
- there exists a feature tracker which can, given

the image sequence, solve the frame to frame correspondence problem and track points over extended periods of time.

- each object in motion yields ≥ 3 points which are tracked over the entire image sequence,
- a motion must have a nonzero rotational component.

Again, these are minimal assumptions. In general performance will be better with more frames and more points taken over a wide change in rotational angles.

2.2.1 Input representation: tracks of shapes in motion

The input to the factorization procedure is a matrix I, representing tracks, i.e. image positions of feature points over time. We assume there are p feature points over f frames, and that in frame i point j is at pixel location $(u_{i,j}, v_{i,j})$ in the image plane. These can be interpreted as a pair of matrices U and V giving the horizontal and vertical component of the point's location respectively. Let U' and V' represent the same position matrices where each row has been shifted so as to have *mean 0 (i.e. independently subtract the centroid* of each frame from every element of that row). It is not necessary that every frame of the motion sequence be in the input matrix. In particular, while dense sampling may be necessary for tracking, the input matrix can be made from significantly fewer frames.

In the 2D case, our input matrix is $f \times p$ and is simply U. For the 3D case we assume an input matrix W which is $(2f) \times p$ with $W = \begin{matrix} U' \\ V' \end{matrix}$. If the dimensionality of the input does not matter, we will use I to represent the input track matrix of size $m \times p$.

2.2.2 Representing Shape

We represent a point in the world coordinate system as ρ_i. For the projection of this point in the 2D case, we adopt a homogeneous representation $(x_i, z_i, 1)$, so we can represent translation. In 3D, where translation is removed by subtracting the centroid of each frame, we use the representation (x_i, y_i, z_i). We can then collect p of these point into a shape matrix, say S of size $3 \times p$, by considering each point as a column in the matrix.

2.2.3 Representing Motion

In 2D let α_i be the angle between the X axis and the camera in frame i. The projection of point ρ_j into the image is given by $u_{i,j} = [\cos(\alpha_i), \sin(\alpha_i), t_i] \cdot [x_j, z_j, 1]^\top$, where t_i is the projection, onto the frame f_i, of the translation vector (as measured from the first frame). We can collect this into a motion matrix, M_{2d}.

In the 3D representation, we assume a fixed world coordinate system. In this coordinate system let $\vec{r}_i$ be a unit vector (represented by its endpoint) which is aligned with the image rows in frame f_i, and let $\vec{c}_i$ be the unit vector aligned with the image columns in frame f_i. Given this, we see that $U'_{i,j} = \vec{r}_i \cdot p_j$ and $V'_{i,j} = \vec{c}_i \cdot p_j$. Thus we can build a $(2f) \times 3$ motion matrix taking $\vec{r}_i$, $i = 1..f$ as the first f rows and $\vec{c}_i$, $i = 1..f$ as rows $(f+1) \ldots 2f$.

With this notation we have $U = M_{2d}S$ and $W = M_{3d}S$, or letting the dimension be implicit $I = MS$.

2.2.4 Factoring the input matrix

Let us consider the SVD of I, and let $\hat{\Sigma}$ be the submatrix of Σ containing the numerically nonzero singular values. Let $\hat{L}$ ($\hat{R}$) be the associated rows (columns) of L (R respectively). Then, as is always true for the SVD, $I = \hat{L}\hat{\Sigma}\hat{R}^\top$. We drop the $\hat{}$ for simplicity and henceforth the exact interpretation of L Σ or R will be given by the context.

Note that the dimensionality of the L and R matrices are exactly the same as the dimensionality of the shape and motion matrix because if the motion/shape is not degenerate, then $\mathcal{R}(I) = 3$. If we let $M' = L \cdot \Sigma^{\frac{1}{2}}$ and $S' = \Sigma^{\frac{1}{2}} \cdot R^\top$, then we even have the same form $I = M' \cdot S'$. In fact, these matrices are, up to an affine transformation, exactly the shape and motion matrices, i.e. there exists a nonsingular A such that $M = M' \cdot A, S = A^{-1}S'$. The exact procedure for obtaining A depends on the dimension as detailed in [Tomasi and Kanade, 1990c] and [Tomasi and Kanade, 1991]. Intuitively, A is obtained by requiring the resulting motion matrix to have the proper orthogonality properties.

3 Description of factorization-based segmentation of multiple motions

In this section we first establish a relationship between the number of singular values and the

number of motions. We present a simple example of the approach on real data. We end this section with a more technical discussion of the determination of rank and our cluster analysis process.

3.1 Representing multiple motions and the related number of singular values

When there is a single motion we consider the same representation as in [Tomasi and Kanade, 1990c] and [Tomasi and Kanade, 1991], (see Section 2.2) which results in a matrix formulation of $I = MS$, where I is $m \times p$, M is $m \times 3$, S is $3 \times p$ and $m = 2f$ or f depending on the dimension.

We now consider the case where there are two motions. Let M' and M'' be motion matrices of size $m \times 3$. Let the associated shape points be S', S'' consisting of p' and p'' points respectively. Assume these points are associated (in some order) with tracks $t_i, i = 1, \ldots, (p' + p'')$.. Then we can represent the track matrix $I = \mathcal{M}\mathcal{S}$, where $\mathcal{M} = [M'|M'']$ with | being matrix concatenation. Let each column $\mathcal{S}_j$ be of the form $[S'_{j,1}, S'_{j,2}, S'_{j,3}, 0, 0, 0]^\top$ if track t_j is associated with a point S'_j and be of the form $[0, 0, 0, S''_{j,1}, S''_{j,2}, S''_{j,3}]^\top$ if track t_j is associated with point S''_j.

That such a decomposition accounts for the track matrix is easily shown. Generalizing this to N motions is straightforward, resulting in a motion matrix $\mathcal{M}$ being $m \times 3N$ and the shape matrix being $3N \times \sum_{k=1}^{k=N} p_k$.

3.1.1 Multiple motions and factorization: Definitions and observations

In some ways the motion matrix M is, in reality, not a motion at all. It is not an optic flow; it is not a parametric equation defining the path of the object; it is not even, necessarily, the path taken by any point in the scene. Instead, each row in the motion matrix can be interpreted as the transformation which takes a world point, in a fixed frame of reference, to its projection in the associated image frame. A motion is then a subspace of f-dimensional space defined by the span of the columns of M. The path (i.e. $x(t), y(t), z(t)$) taken by any single object point is represented as a single point in this f-dimensional space, and hence we call this *path space*. A rigid body motion requires all points being considered to follow paths which are within the span of the columns of the associated motion matrix M. Since we cannot directly measure M, we will generally consider the information in the columns of I, to define the observable motion. We will use motion to mean either real motion (M) or observable motion (I) where the context should disambiguate the interpretation. We now explore the ramifications of this representation.

In what follows, let $I_i, i = 1..N$ be track matrices of different motions with $\mathcal{R}(I_i) > 0$. Let M_i be the associated motion matrices and S_i the shapes. Assume no noise so that $\mathcal{R}(I) = Rank(I)$.

Definition 1 *A track matrix I corresponds to a motion if and only if (hereafter* **iff** *) its columns span a nonempty subset of path space*

Observation 1-1 *A track matrix I corresponds to a motion* **iff** $\mathcal{R}(I) > 0$.

Definition 2 *Assume I_1 is associated with a single motion (M_1 and S_1 assumed to be $f \times 3$ and $3 \times p$ respectively.) I_1 is called nondegenerate* **iff** $\mathcal{R}(I_1) = 3$. *That is, a nondegenerate motion spans a 3 dimensional subspace of path space. (Similarly define nondegenerate M_1 and S_1)*

Observation 2-1 *For N motions we have $\mathcal{R}(M) \leq 3N$, $\mathcal{R}(S) \leq 3N$ and hence it follows that $\mathcal{R}(I) \leq 3N$.*

Observation 2-2 *A nondegenerate track matrix is the product of a nondegenerate motion and nondegenerate shape. Nondegenerate motions require at least 3 frames. Nondegnerate shapes requires at least 3 points.*

Definition 3 *The track matrix I_2 contains a different motion from that of I_1* **iff** *the span of the columns of I_2 is not contained within the span of the columns of I_1. That is I_2 has a motion different from I_1* **iff** *there are points in path space which could be associated with I_2 which could never be generated by the motion underlying I_1.*

Observation 3-1 *It follows that I_2 contains a motion different from that of I_1* **iff** $\mathcal{R}(I_1|I_2) > \mathcal{R}(I_1)$.

Definition 4 *Two motions I_1 and I_2 are said to be different motions* **iff** *they each contain a motion different from the other. (Note that if the span of I_1 is a proper subset of the span of I_2 then I_2 contains a motion different from I_1, but not the other way around).*

Observation 4-1 *Obviously I_2 and I_1 are different* **iff** $\mathcal{R}(I_1|I_2) > \max(\mathcal{R}(I_1), \mathcal{R}(I_2))$.

Observation 4-2 *Note that such definitions, in terms of subspaces, may not totally capture the intuitive notion of "different motions". If there are two well separated clusters within a single subspace, humans might interpret them as different motions. This definition is saying that two motions are the same if there exists a rigid body interpretation that makes them the same. It does not preclude using additional information to further label submotions with different labels. (In fact, the algorithm to be presented can often "segment" these well separated clusters, however it knows that they are, according to the above definitions, 2 clusters from the same motion.)*

Observation 4-3 *Adding a single track from a "different" motion to I must increase the rank of I by 1.*

Observation 4-4 *If track matrix I is decomposed into 2 parts I_1 and I_2, then $\mathcal{R}(I) \leq \mathcal{R}(I_1) + \mathcal{R}(I_2)$.*

Definition 5 *I_1 and I_2 are called linearly independent motions* **iff** $\mathcal{R}(I_1|I_2) = \mathcal{R}(I_1) + \mathcal{R}(I_2)$.

Observation 5-1 *A track matrix from N linearly independent motions will have $\mathcal{R}(I) = 3N$.*

Observation 5-2 *If a track matrix I, composed of N linearly independent nondegenerate motions (containing at least 4 points each), is decomposed into 2 parts I_1 and I_2, then $\mathcal{R}(I) = \mathcal{R}(I_1) + \mathcal{R}(I_2)$* **iff** *for every set of shape points $\{S\}_i$ associated with a single motion $\{M\}_i$ in I, the tracks from these points/motions are contained entirely in either I_1 or I_2. In simpler terms, under general motion assumptions the segmentation of I is good (but not necessarily complete) if and only if the number of nonzero singular values before segmentation is the same as the sum of the nonzero singular values of the segments.*

To prove this last observation, let $\{M_iS_i\}_j$ be the tracks from motion i that appear in segment j, with the understanding that if no points of motion i appear in segment j, then $\{M_iS_i\}_j$ is empty. Also let segment $j = 0$ refer to the original (unsegmented data). By definition we have, modulo some column permutations, $I_j = \{M_1S_1\}_j|\ldots|\{M_NS_N\}_j$, which implies $\mathcal{R}(I_j) = \sum_{i=1}^{i=N} \mathcal{R}(\{M_iS_i\}_j)$. Thus we can restate our observation as

$$\sum_{i=1}^{i=N} \mathcal{R}(\{M_iS_i\}_0) \leq \sum_{i=1}^{i=N} \mathcal{R}(\{M_iS_i\}_1) + \sum_{i=1}^{i=N} \mathcal{R}(\{M_iS_i\}_2) \quad (1)$$

with equality holding if and only if the segmentation is good. That a good segmentation implies equality follows directly from the above equation and the definition of linear independence. To show that equality implies a good segmentation we argue as follows. First, note that since the number of nonzero singular values is always nonnegative, the splitting of any motion cannot reduce the right hand sum. Since each motion contains at least 4 points either all of its points are in one segment, in which case it contributes a value of n to the right hand side, or its points are split across the partition and it contributes a minimum of $n + 1$ to the right hand sum. Noting that each motion's contribution to the left hand side is exactly n, the observation follows.

One of the reasons for proving the observation as we did is that it provides insight to what may happen if the assumptions of the observation are violated. If there are motions with only 3 points, then a segmentation may fail in an undetectable manner. More of a potential problem is that if there are motions which are linearly dependent, then there may exist incorrect segmentations into two groups which will *not* increase the total number of singular values. Luckily, the observation often holds even if there are linearly dependent motions. If some of the motions are dependent, the equality will hold if and only if all the points in the linearly dependent subset of motions are grouped into a single partition.

The importance of this final observation should not be overlooked. It provides a theoretical basis for checking the segmentation. If we start with $3N$ singular values and eventually get N subsets with 3 singular values each, we know that we have a correct segmentation!

3.2 Overview of approach

The method can be summarized as the following steps:

1. Find tracks (we do not address this issue in this paper) and compute track matrix I
2. Compute SVD yielding L, Σ, R
3. Determine $\mathcal{R}(I)$ and prune columns and rows of L and $R^\top$ respectively.
4. Generate clusters, using R, to get potential segmentations.
5. For a potential segmentation, remove the 'segmented' tracks from original image yielding I_i, and compute the SVD of each separately. If $\mathcal{R}(I) = \sum_i \mathcal{R}(I_i)$, then we know the partition is good, otherwise we try the next clustering at this level. If the last clustering fails, backtrack to the call to this level.
6. Given a good partition, if a cluster has ≤ 3 singular values we compute shape and motion, else treat this as a new input matrix for the next level of segmentation and recursively goto step 2.

The first step, which we assume is handled by some other process, determines the tracks of points in the scene. We also expect an error estimate for the track information. It is not really important that the tracks are dense in either time or space, although the quality of the motion estimate and shape estimates depend on those densities respectively.

3.3 A simple 2D Example using Real Data

We now present a simple example which we will follow through as we describe the algorithm in more detail. The motion was obtained by moving a camera in a plane using a precision Datel rotation stage while keeping a high contrast scene in view. While the camera rotated, 2 objects in the scene were independently moved. One object (on the right) was rotated about its axis, and the other was translated with a small amount of local rotation. We obtained our epi-image by grabbing one 512 pixel scanline per frame time for 100 frames. To make the segmentation task more difficult we moved the epi's of the two "objects" closer to produce the epi-image and its associated edges shown in Figure 1. The high contrast objects allowed for easy "tracking"

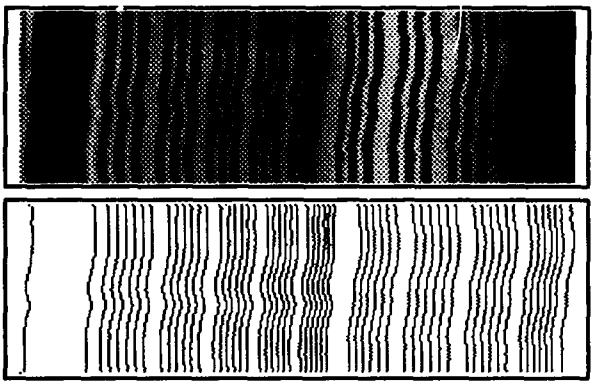


Figure 1: Top shows original epi-polar image (graylevel camera scanlines). The bottom figure shows the tracks found for 57 points over 101 frames.

of features using a simple Sobel edge detector and edge linking which locally fits a quadratic to the Sobel response and tracks the maximum with subpixel precision. Tracks which were not continued from the first line to the last line were not included in the input matrix. Figure 2 shows the segmented tracks determined by the algorithm.

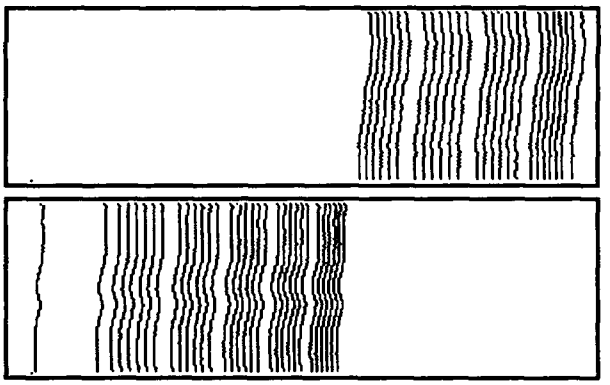


Figure 2: Segmented tracks from detected motions

3.4 Some Intuition as to how/why it works

To help the reader get a better feel for what the components of the SVD are, we present some partial reconstructions and component analysis.

By examining Figure 3, one can get a feel for which components of the motion are associated with each singular value. The scene has 20 frames of 20 points (ten per motion) embedded into a 20×256 image. The singular values associated with the input are 31.321, 3.724, 1.934, 0.284, 0.131, 0.000444, 0.000000, 0.000000, ... The figure shows the incremental effects of reconstructing the tracks

using the SVD with only the first singular value (all others set to zero), the first and the second, and finally all three. In the image which uses only the first singular value (second from top) we can see that the leftward "flow" of the underlying the tracks is captured. Including the components associated with the second singular value (third from top) adds some of the right hooking but still does not capture everything. Adding in the third would result in an image indistinguishable from the original and is not shown. The last image (bottom) shows one of the two motion segments recovered (it is exact). The grouping for segmentation can be found in the first right vector, or in other words, the first principal component.

3.5 Determining rank

A central part of our segmentation algorithm requires determining the rank or number of nonzero singular values for the idealized input which lead to the measured input matrix I. If, as is usually the case, our desired input matrix, say D, is perturbed by noise (error), say E, then we cannot expect the singular value decomposition of $I = D+E$ to yield the exact number of singular values of D; i.e. we need to do more than determine the number of nonzero singular values. In determining an approximation to the rank of D we will use some knowledge of E.

It can be shown that the difference between the singular values of the ideal input, $\sigma_k(D)$ and the singular values of the measured input, $\sigma_k(I)$ satisfy the following properties:

$$|\sigma_k(I) - \sigma_k(D)| \leq \epsilon\sigma_1(I), \quad \forall k \leq p, \qquad (2)$$

$$|\sigma_k(I) - \sigma_k(D)| \leq \sigma_1(E) < \|E\|_F \quad \forall k \leq p, \quad (3)$$

and

$$\sum_{k=1}^{p}(\sigma_k(I) - \sigma_k(D))^2 \leq \|E\|_F^2 = \sum E_{i,j}^2 \qquad (4)$$

where $\|\cdot\|_2$ and $\|\cdot\|_F$ are the second and Frobenius matrix norms respectively, and ϵ is the machine precision for computation. See [Golub and van Loan, 1983, Sect 6.5 and Cor. 8.3.2, 8.3.5] for proofs and more detail.

We now develop bounds on $\mathcal{R}(i)$. Let be k^* be the smallest integer such that $\forall k \geq k^*$, $\sigma_k(D) = 0$.

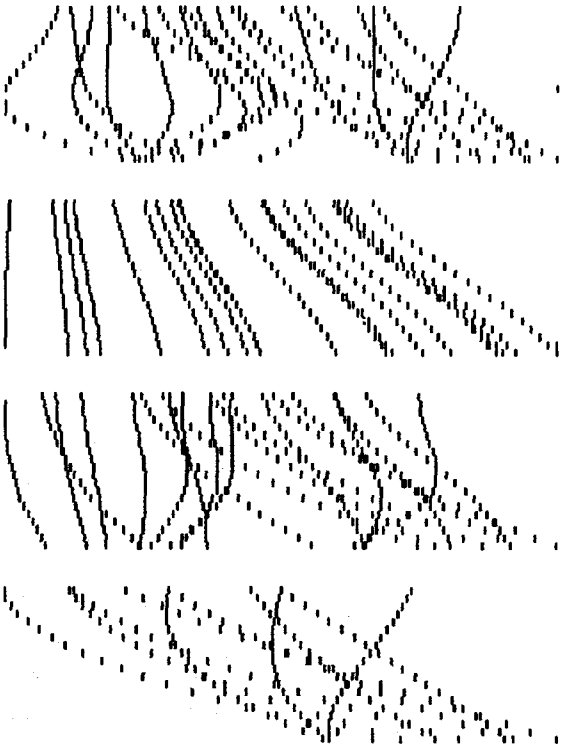


Figure 3: Partial SVD reconstruction of 2 synthetic 2D motions. The first image shows scaled version of original track image. The second shows partial reconstruction of the original tracks using first singular value, i.e. $l_1 \otimes r_1$. The third shows the partial reconstruction using the first two singular values. Adding the third singular value gives back the original image. The final image shows the segmented and reconstructed tracks of one of the motions. Note the segmentation is exact. (See text for more details.)

Notice for $k \geq k^*$ we have $\sigma_k(I) - \sigma_k(D) = \sigma_k(I)$. Combining this observation with equations 2– 4 we have a lower bound

$$\mathcal{R}(I) \geq \max\ k \text{ s.t.} \begin{cases} \sigma_k(I) \geq \epsilon\sigma_1(I), \\ \sigma_k(I) \geq \|E\|_F, \text{or} \\ \sqrt{\sum_{j=k}^{j=p} \sigma_j^2(I)} > \|E\|_F \end{cases} \qquad (5)$$

These lower bounds on $\mathcal{R}(I)$ require a conservative estimate (over estimate) of $\|E\|_F$. Note this relies on relatively weak knowledge of the noise E, and makes no assumption about the shape of the error distribution. In the examples in this paper we assume $\|E\|_F \leq \delta_1$, where δ_1 is set to be greater than

or equal to the expected point RMS error. For real datasets, the expected RMS is computed by computing the SVD of a known nondegenerate single motion and subtracting the reconstruction (using first 3 singular values) from the original track matrix. For the example images estimated RMS was .04.

In our upper bound we assume that there is a computable predicate $Np(E)$, which when applied to a matrix E returns true only if E "must" be considered pure noise. As we will see later, because we are interested in tracks of motion, it is quite reasonable to assume that any matrix must be noise if $|E_{i,j}| < \delta_2, \quad \forall i, j$. For the real examples we have set a very conservative estimate of $\delta_2 = .0016$. (Our upper bounds would be smaller if we were less conservative). Another reasonable noise predicate can be obtained by assuming that for all observable I we will have $\|E\|_F \geq \delta_2$.

Assuming that the predicate $Np(E)$ is conservative (i.e. it never returns TRUE if the matrix E could be valid data), we can get an upper bound on $\mathcal{R}(I)$:

$$\mathcal{R}(I) \leq \min \; k \;\; \text{s.t.} \; Np\left(I - \sum_{i=1}^{i=k} \sigma_i l_i \otimes r_i^{\top}\right) \quad (6)$$

In words, the upper bound is the smallest k such that the difference between I and the kth reconstruction of I must be noise.

It should be noted that the sanctity of these "bounds" depends on the conservative measures of the noise models, while the distance between them (and hence the usefulness in approximating $\mathcal{R}(I)$) depend on the error model being as sharp as possible, i.e. not too conservative. In most cases the upper and lower bounds differ by more than 1 and hence only restrict $\mathcal{R}(I)$, rather than determining it. Because, as we shall see later, it is convenient for our algorithm to have a single number for $\mathcal{R}(I)$, in these cases we make a final "heuristic" determination of it using a knee finding technique on the logarithm of singular values between the upper and lower bound. That is, we chose as $\mathcal{R}(I)$ the i that has maximal curvature on the curve $(i, \log(\sigma_i))$ for i in the range determined by equations 5– 6. This last heuristic step has correctly determined $\mathcal{R}(I)$ in most of our test cases. It is important to note that without the bounds to determine the search window, the knee finding would be much more difficult since in the region of valid singular values there may be other "local" knees, especially in the case where there are multiple motions of significantly different magnitudes.

For our 2D example, the 8 largest singular values were 13820.38, 51.91 16.24 3.51, 1.96, 1.26, 1.14, and .96. Just looking at the above numbers it would be hard to determine if there was one or two motions without knowledge of the expected noise and the bounds on $\mathcal{R}$. The algorithm for determining $\mathcal{R}$ determines the bounds 4 and 34 and the knee finder determines that there are 5 nonzero singular values. After segmentation the algorithm found one cluster had 3 nonzero singular values, the other 2 nonzero singular values.

To give you an idea of the complexity of the task of determining the number of significant singular values, consider the example shown in Figure 4. Here we have tracks over 40 frames of four motions with 10 points each. The lowest solid line shows the singular values without noise. In this case the $\mathcal{R}(I)$ is obvious since the nonsignificant singular values are numerically equal to zero (and hence cannot be plotted here since the plot is logarithmic). The other lines show the singular values when white noise is added with standard deviation .1, .2.,3 .4 and .5. Notice there are several "knees" and when the noise level is too high the last knee becomes indistinguishable from noise. If we did not have bounds to limit the search for $\mathcal{R}(I)$ it would be very difficult to determine the correct number of significant singular values.

3.6 Discussion of cluster analysis

First, we point out that the right vector associated with the largest singular value, call it, r_1, need not be the vector which gives the best segmentation. If the two motion parameters are sufficiently mixed together in a particular dimension, but separated in another, the row associated with a smaller singular value may actually give a cleaner segmentation. For a large number of motions it is often the case that a row r_i will allow one to easily segment out a single motion, while lumping the remaining $N-1$ motions into a single cluster. Therefore, it is useful to consider numerous r_i's.

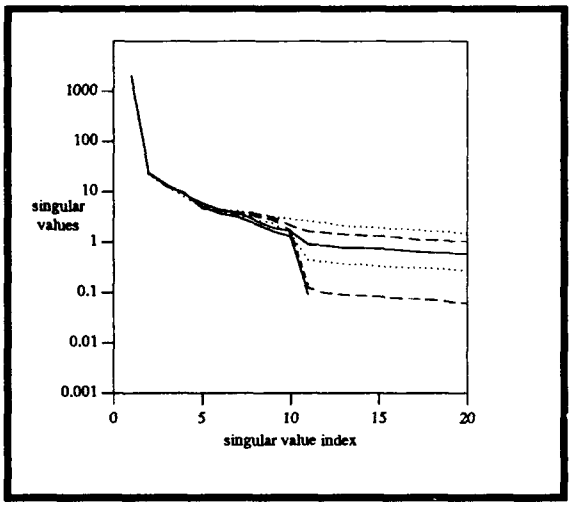


Figure 4: Here we see the singular values for a synthetic 2D case as the standard deviation of additive white noise is increased from 0 to .5. There are 4 motions with 10 points each tracked over 40 frames.

This is really a general clustering problem, closely related to clustering used in factor analysis. However, because we have the ability to check a partition of the data, we take a slightly more conservative approach. Our goal is not to find all clusters at once, but rather to proceed by breaking the data up into 2 partitions and checking the break. If it is good, then we recursively solve each of the sub-problems. Thus there are three parts to our clustering algorithm: initialization of clusters, refinement of clusters, splitting data and checking clusters.

We perform the clustering as follows. First we use the first right vector and use this to break the track matrix into two new input matrices. The SVD of each of the proposed segments is computed and we check the number of singular values as suggested by Observation 5-2. If this check is satisfied we continue segmentation on the new input matrices. If it fails, we then try clustering on the second right vector. If this fails we try the sum of the first two vectors weighted by their singular values. Obviously we could add other choices, but these have been sufficient for now.

For the 1D clustering problem initialization is straightforward. We know we are looking for 2 clusters, and we use the cluster with maximal separation for initialization. In particular, we compute the nearest neighbor distance for each point and use the largest distance as the breakpoints between the clusters. For the 2D clustering, we initially cluster on the maximal separation of the first right vector.

Given the initial clusters, we refine them attempting to minimize the RMS distance of each point from the cluster center. An algorithm for this is detailed in [Duda and Hart, 1973, Ch. 6]. Applying this iterative algorithm results in 2 clusters and a measure of compactness (root mean square distance to cluster center) for each cluster. In most of our examples, the number of iterations =1, i.e. the initial clustering is already minimal. In the very noisy cases however, the initial clustering is a poor approximation and the iterative improvement offered by this approach can significantly improve the quality of segmentation. The squared error nature can, at times, cause a few outliers to be included in the wrong cluster. The issues of normalization in clustering arise here (e.g. do we use $\sigma_i r_i$, $\sqrt{\sigma_i} r_i$, or simply use r_i), see [Duda and Hart, 1973, pp. 216 ad pp. 224]. Currently we do not normalize.

Figure 5 shows the first two components of R used in clustering on the 2D example. The points are shown with labels so you can see which points correspond to which motions. Note that the first right vector (associated with the first singular value) correctly classifies the motions resulting in the aforementioned segmentation.

4 Initial Experimentation

Our initial experiments have been very promising. Figure 6 illustrates our method applied to a dataset with tracks from four different motions. The epi-image was constructed in a similar fashion to our previous 2D example with real data except that additionally, we combined epi-images acquired at different time periods in order to have an example with more motions. The tracks from each motion are contiguous and can be distinguished using the bottom image in this figure. Here the tracks from two different motions can be clearly discerned; the tracks not in this image, make up the other two motions. In Figure 7, we show the

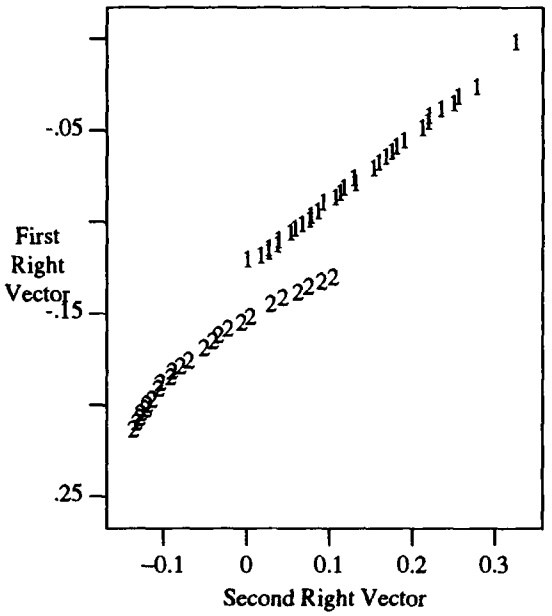


Figure 5: Here we see the clustering for each track shown in Figure 1 using its entry in the first right singular vector as the Y coordinate, and its entry in the second right vector as the X coordinate. Each point is labeled with the motion which generated it.

results of clustering using the 2nd right right vector. Our algorithm segmented the p' ints marked with "x" from those with the "o". The latter correspond to the motion on the right in the bottom image of Figure 6. However, it is clear from this plot, that it is possible to segment all four motion components just using the first two right vectors of the original SVD, without further recursion.

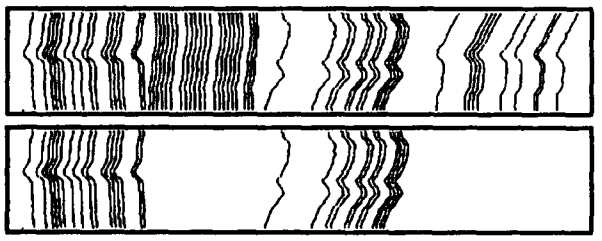


Figure 6: Top shows tracks found for 59 points (4 motions) over 100 frames. The bottom shows the tracks of two clearly distinguishable motions.

We have also experimented with synthetic data using additive noise, $N(0, sd)$, displacing each track point. We have been able to segment two motions with complete accuracy for noise levels up

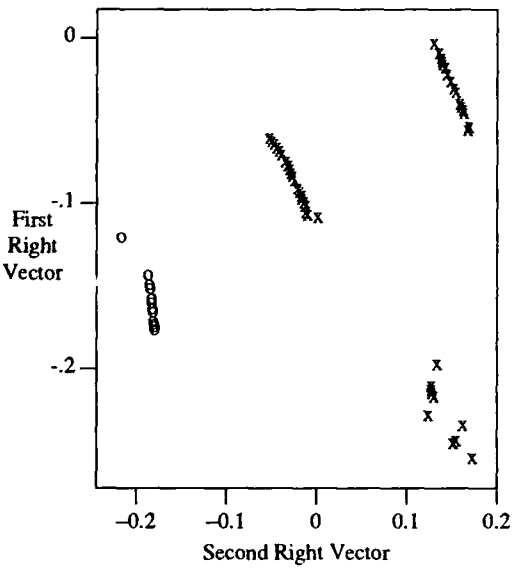


Figure 7: Here we see the clustering of the first 2 right singular vectors for a realistic case with four motions. The first segmentation splits the tracks labeled with an "x" from those with a "o" - the latter correspond to the motion given by the tracks shown on the right in the bottom of Figure 4.

to 50% of the point value. The results of segmenting a difficult 3D synthetic example with three motions can be seen in Figure 8. (Points are labeled with their corresponding motion.) The figure shows a plot of the first two right singular vectors for each of 3 motions (10 random (overlapping) shape points per motion, 50 frames of motion). The first pass of the algorithm split the motion into two groups, one containing motion 1 and the point labeled 3', and the second the remaining points. Initially it found 9 singular values, and after segmentation there were 4 and 6 singular values. The group with 6 singular values (motions 2 and 3) were further decomposed into their correct motion components.

This example also shows why it might be useful to disregard the cluster labels for points on the fringes of the cluster and then add them into the cluster after recomputing the SVD. One way to do this would be to split the tracks conservatively into two groups I_1 and I_2 (more than $\mathcal{R}(I)$ points each), and compute the SVD of each. Then for any track t_i that was on the fringe, compute $w = \hat{L}_k^\top t_i$. This should have $w_i \approx 0, \forall i > \mathcal{R}(I_k)$ if t_i is in the

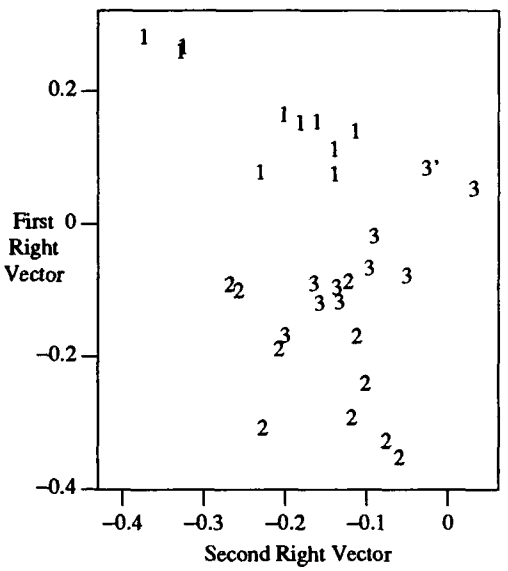


Figure 8: Here we see the clustering of the first 2 right singular values for 3 synthetic motions. Point 3' is mistakenly clustered in with the data from motion 1.

same subspace. While it sounds promising, this technique has yet to be implemented and tested.

5 Critical analysis

While the technique presented in this paper does, in our opinion, a very good job at segmentation of multiple motions, there are still a few difficulties with the approach. Most of the difficulties are actually problems with the underlying factorization technique, and may, we hope, be overcome with additional effort.

We present a check-list of the advantages (+) and disadvantages (−) of this approach. Those aspects which are both pros and cons will be marked with ±.

+ The segmentation approach using the R component of the SVD appears to be extremely powerful.
+ The factorization technique simultaneously provides shape and motion.
+ Shape is represented relative to object centroid and hence is stable with respect to short-baselines in motion.
+ A technique for bounding and approximating the numerical rank $\mathcal{R}(I)$ has been developed
+ The segmentation approach comes with a theoretically derived check on the quality of the segmentation (assuming linearly independent motions and that $\mathcal{R}(I)$ is correct)).
± The segmentation method is quite robust w.r.t. positional noise.
± Cost is $O(mn^2 + n^3)$ (with a reasonable constant). If the number of points and frames are not too large this is quite reasonable. For example for 200×100 3D data set (hence a 400×100 input matrix) the approach takes $\approx$ 64 CPU seconds on a 12Meg-SparcStation1.
− The representation assumes orthographic projection.
− The segmentation method can have difficulties when there are nearly dependent motions.

6 Conclusions and Future Work

This paper generalized the factorization approach for simultaneous recovery of motion and shape to handle multiple motions. It presented a technique, using the information available from factorization, for the segmentation of multiple motions. The method includes a way to bound the numerical rank of the input and proves how to use this to check, in general, that the segmentation is valid. The segmentation technique was demonstrated on numerous real and synthetic examples.

There is considerable future work to be done in the area of factorization-based motion analysis. For example, consider the many technical reports on which Tomasi and Kanade appear to be working [Tomasi and Kanade, 1991]. Much of that future work applies here as well. There are a few things on which we expect to continue working as they have particular impact on our segmentation approach. These areas include a more thorough error analysis of the segmentation including a better analysis of different approaches to clustering (including removing the fringe points), comparison with previous motion segmentation techniques, and extending the factorization approach to handle partial tracks and tracks distorted by perspective.

Acknowledgments

This work was supported in part by Darpa Contract #N00039-84-C-0165 and in part by the National Sci-

ence Foundation Grant PYI award #IRI-90-57951 with additional support from Seimans and A&TT.

References

[Adiv, 1985] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE PAMI*, 7(4):384–401, July 1985.

[Bergen *et al.*, 1990a] J. Bergen, P.J. Burt, R. Hingorani, and S. Peleg. Computing two motions from three frames. In *ICCV*, pages 27–32. IEEE, Dec 1990.

[Bergen *et al.*, 1990b] J. Bergen, P.J. Burt, R. Hingorani, and S. Peleg. Transparent-motion analysis. In *Springer Lecture Notes in Computer Science: Proceedings of ECCV*, volume 427, pages 566–569. Springer-Verlag, April 1990.

[Dickmanns, 1989] E.D. Dickmanns. Subject-object discrimination in 4d dynamic scene interpretations for machine vision. In *Proc. IEEE Workshop on Visual Motion*, pages 289–305. IEEE, March 1989.

[Duda and Hart, 1973] R. O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* Wiley, NYC, NY, 1973.

[Fennema and Thompson, 1979] C. Fennema and W.B. Thompson. Velocity determination in scenes containing several moving objects. *CVGIP*, 9:301–316, 1979.

[Golub and van Loan, 1983] G.H. Golub and C.F. van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, MD, 1983.

[Heita and Bouthemy, 1990] F. Heita and P. Bouthemy. Motion based segmentation. In *Proc. Int. Conf on Pat. Rec.*, pages 378–383. IEEE, June 1990.

[Murray and Buxton, 1987] D.W. Murray and B.F. Buxton. Scene segmentation from visual motion using global optimization. *IEEE PAMI*, 9(2):220–228, Mar 1987.

[Peleg and Rom, 1990] S. Peleg and H. Rom. Motion based segmentation. In *Proc. Int. Conf on Pat. Rec.*, pages 109–113. IEEE, June 1990.

[Press *et al.*, 1988] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C.* Cambridge Univ. Press, Cambridge, UK, 1988.

[Shizawa and Mase, 1990] M. Shizawa and K. Mase. Simultaneous multiple optical flow estimation. In *Proc. Int. Conf on Pat. Rec.*, pages 274–278. IEEE, June 1990.

[Subrahmonia *et al.*, 1990] J. Subrahmonia, Y.P. Hung, and D.B. Cooper. Motion based segmentation. In *Proc. Int. Conf on Pat. Rec.*, pages 390–398. IEEE, June 1990.

[Tomasi and Kanade, 1990a] C. Tomasi and T. Kanade. Shape and motion from image streams: a factorization method. Technical report, Carnegie Mellon Computer Science, Pitt. PA, Sep 1990. TR Num. CMU-CS-90-166.

[Tomasi and Kanade, 1990b] C. Tomasi and T. Kanade. Shape and motion without depth. In *Proceedings of the DARPA Image Understanding Workshop*, pages 258–271. DARPA, Morgan Kaufman Pub. San Mateo CA, Sep 1990. ISBN 1-55860-140-6.

[Tomasi and Kanade, 1990c] C. Tomasi and T. Kanade. Shape and motion without depth. In *ICCV*, pages 91–95. IEEE, Dec. 1990.

[Tomasi and Kanade, 1991] C. Tomasi and T. Kanade. Shape and motion from imgage streams: a factorization method. 2. point features in 3d motion. Technical report, Carnegie Mellon Computer Science, Pitt. PA, January 1991. TR Num. CMU-CS-91-105.

[Yamamoto, 1990] M. Yamamoto. A segmentation method based on motion from image sequence and depth. In *Proc. Int. Conf on Pat. Rec.*, pages 230–233. IEEE, June 1990.

Recursive Multi–Frame Structure from Motion Incorporating Motion Error

J. Inigo Thomas and J.Oliensis*
Computer Science Department
University of Massachusetts at Amherst
Amherst, MA 01003

Abstract

Recovering structure from motion even using information from multiple image frames is difficult, in part because motion error can introduce large, correlated errors in the structure estimate. A method is proposed for recursively recovering structure from motion that can deal with this problem. Encouraging results on real images and synthetic data are presented.

1 Introduction

Two frame structure from motion is known to be inaccurate. The reasons for this are familiar: for two frames, each structure value is determined by a single measurement which is unreliable for distant points, and which is strongly affected by small errors in the motion. The natural remedy to this problem is to refine the structure estimation by combining measurements from many frames [2] [3] [8] [11] [12] [13] [16]. In earlier work [14] [18], we have presented an implementation of this procedure which differs from previous approaches in that the *motion error* is explicitly taken into account. In this paper, a detailed theoretical derivation of the algorithm is presented, together with new experimental results on a real image sequence.

The difficulty of multiframe structure from motion has several sources. First, even assuming the motion is known, determining the structure of a pcint from two noisy images is a non–linear problem, and biased in the depth estimate. This problem was studied experimentally in [17]; the results indicate that in fact the non–linearity of the pure structure measurement is *not* a serious problem—it can be compensated for by a multiplicity of measurements from different camera positions.

Another, probably more serious, difficulty is the *motion error*. It is known that small errors in the rotation can introduce relatively large errors in depth measurements [5]. Moreover, motion error introduces strong *correlations* in the structure errors. For instance, if the translation is identified incorrectly, then the measured positions of all points will be displaced away from their actual ones in approximately the same direction: their position errors will be correlated. An analogous result holds for rotation error. Conversely, *the correlations in the structure errors are the record of the motion error.* In order to include the potentially large effects of motion error, and to compensate for them, a record of these correlations must be maintained in the current error estimate. This information can then be used by a recursive algorithm for updating the old structure estimate based on new image information. This explicit inclusion of motion error is the main new contribution of this paper.

2 Overview of the Algorithm

The camera is assumed to be navigating in an unknown, fixed environment, consisting of isolated 3D points. The 2D frame–to–frame correspondences of these points are assumed given. The goal is to recursively determine the locations of the 3D points.

An obvious strategy is, for each new image, to combine the old structure estimate with the information contained in the new image, weighted by their respective uncertainties. However, this *shape from pose* approach is potentially unstable, since initially the structure estimate is quite inaccurate. Instead, at each iteration, Horn's relative orientation algorithm [9] is used to determine a new structure estimate from the most recent *pair* of images, which is then fused with the previous estimate. Since Horn's algorithm gives a relatively robust structure measurement, this approach can recover from errors in the initial cumulative estimate more easily than the shape–from–pose method. Also, because of the simplicity of Horn's objective function, an approximate error analysis is relatively tractable. However, other 2 (or 3) frame motion algorithms could be used equally well.

The algorithm is as follows. At each time step, Horn's algorithm is used to recover the structure and relative orientation for the most recent image pair. Also, the structure error of this algorithm is estimated, consisting of the complete covariance matrix including cross–correlations between different 3D points. These cross–correlations represent the effects of the uncertainty in the relative orientation, i.e. the frame–to–frame motion error, as stated above.

The output of Horn's algorithm is then fused with the previous structure estimate. To do this, both the old

*This work was supported by DARPA (via TACOM), contract number DAAE07-91-C-R035.

and new structure estimates, and their estimated errors, must be transformed into a common coordinate system. This transformation is not exactly known and induces additional error. The coordinate system chosen is that of the moving camera; the fused structure is maintained in this system, since this is the relevant information for the robot. The fusing is done by the standard Kalman filter method.

In principle, the structure can be computed with arbitrary accuracy given sufficiently many image frames. More precisely, what can be computed accurately is the absolute structure or *shape* [19] , i.e. the structure in an object–centered, camera–independent coordinate system. On the other hand, the current camera pose, and therefore the depths as measured in the current camera coordinate system, are determinable only with a limited accuracy. Previous recursive algorithms for multiframe structure from motion had limited accuracy of shape or structure recovery, because of the limit set by the pose error. In contrast, because the algorithm described here explicitly takes into account the uncertainty in the camera pose, it is potentially capable of determining the shape with an accuracy greater than that with which the pose or motion is recovered.

3 Estimating the Motion Error

To compute the structure error of Horn's algorithm, the error in relative orientation must be computed as an intermediate step. Horn's algorithm computes the motion between the previous (left) and most recent (right) camera frames by minimising the objective function:

$$E = \sum_i (\mathbf{b} \cdot (\vec{\mathbf{r}}'_{l_i} \times \hat{\mathbf{r}}_{r_i}))^2 \equiv \sum_i E_i^2. \qquad (1)$$

Here the notation of [9] has been adopted. The error in this motion estimate is computed by linearizing, as is standard. Thus, the *derivatives* of the motion parameters with respect to the image coordinates must be calculated. $\mathbf{W}$ is used to represent the five motion parameters: two for the translation direction, and three for the rotation. The translation magnitude is omitted since it cannot be recovered due to the well–known scale ambiguity under arbitrary motion. Also, the image coordinates of the two images are represented by a vector $\mathbf{V}$ of length $4m$, where m is the number of 3D points.

The recovered values of the motion parameters correspond to a minimum of the objective function E, and therefore $\partial E/\partial \mathbf{W} = \mathbf{0}$. After a perturbation in the image coordinates, the perturbed motion parameters recovered by Horn's algorithm minimize the new E. Thus the derivative above is again zero when evaluated at the new values for $\mathbf{V}$ and $\mathbf{W}$: it has not changed in value. This implies that:

$$\frac{\partial^2 E}{\partial \mathbf{W} \partial \mathbf{V}} d\mathbf{V} + \frac{\partial^2 E}{\partial \mathbf{W} \partial \mathbf{W}} d\mathbf{W} = 0 \qquad (2)$$

for the small perturbations $d\mathbf{V}$ and $d\mathbf{W}$. Defining

$$N = \frac{\partial^2 E}{\partial \mathbf{W} \partial \mathbf{V}}, \qquad M = \frac{\partial^2 E}{\partial \mathbf{W} \partial \mathbf{W}} \qquad (3)$$

and assuming M has an inverse, equation 2 can be rewritten as

$$\frac{\partial \mathbf{W}}{\partial \mathbf{V}} = -M^{-1} N \qquad (4)$$

In terms of this $5 \times 4m$ matrix, the motion error covariance is:

$$Cov(\mathbf{W}) \equiv E\{d\mathbf{W} d\mathbf{W}^T\} \simeq \frac{\partial \mathbf{W}}{\partial \mathbf{V}} E\{d\mathbf{V} d\mathbf{V}^T\} \frac{\partial \mathbf{W}^T}{\partial \mathbf{V}}. \qquad (5)$$

Assuming that the image noise at each pixel is independent and has the same standard deviation σ, the covariance of $d\mathbf{V}$ is proportional to the identity matrix, and the linearized estimate of the motion error is the 5×5 matrix:

$$\sigma^2 \frac{\partial \mathbf{W}}{\partial \mathbf{V}} \frac{\partial \mathbf{W}^T}{\partial \mathbf{V}}. \qquad (6)$$

3.1 Determining the M matrix

First, the motion parameters incorporated in $\mathbf{W}$ must be specified. Let $\hat{\mathbf{b}}$ be the unit translation vector, and let $\hat{\mathbf{b}}_e$ be the translation actually recovered by Horn's algorithm. Then an arbitrary $\hat{\mathbf{b}}$ near $\hat{\mathbf{b}}_e$ can be represented as:

$$\hat{\mathbf{b}} = \mathbf{b_T} + \sqrt{1 - |\mathbf{b_T}|^2}\hat{\mathbf{b}}_\mathbf{e}, \qquad (7)$$

where $\mathbf{b}_T$ is in the plane perpendicular to $\hat{\mathbf{b}}_e$. This representation is adequate because for this linearised analysis the motion error is assumed to be small. $\mathbf{b}_T$ is represented explicitly by its projection on two perpendicular axes $\hat{\epsilon}_1$ and $\hat{\epsilon}_2$ in the plane normal to $\hat{\mathbf{b}}_e$.

The rotation R is represented using quaternions, but in a more convenient notation. Let the axis of rotation be $\hat{\mathbf{n}}$, and the rotation angle be θ. Then the unit quaternion is:

$$\hat{\mathbf{R}} = (\cos\frac{\theta}{2}, \hat{\mathbf{n}} \sin\frac{\theta}{2}) \equiv (R_0, \vec{\mathbf{R}}), \qquad (8)$$

with $R_0^2 + \vec{\mathbf{R}}^2 = 1$. The rotation is represented by the three–dimensional vector $\vec{\mathbf{R}}$. The rotation matrix is given by:

$$\vec{\mathbf{r}}'_{l_i} = R(\hat{\mathbf{r}}_{l_i}) = (1 - 2|\vec{\mathbf{R}}|^2)\hat{\mathbf{r}}_{l_i} + 2\sqrt{1 - |\vec{\mathbf{R}}|^2}(\vec{\mathbf{R}} \times \hat{\mathbf{r}}_{l_i}) + 2(\vec{\mathbf{R}} \cdot \hat{\mathbf{r}}_{l_i})\vec{\mathbf{R}}. \qquad (9)$$

The vector $\vec{\mathbf{R}}$ is represented in terms of its projections along three orthogonal 3D directions $\hat{\eta}_1$, $\hat{\eta}_2$ and $\hat{\eta}_3$.

The first partial derivatives are:

$$\frac{\partial E}{\partial(\mathbf{b}_T \cdot \hat{\epsilon})} = 2\sum_i E_i(\hat{\epsilon} - \frac{\hat{\epsilon} \cdot \mathbf{b}_T}{\sqrt{1 - |\mathbf{b}_T|^2}}\hat{\mathbf{b}}_e) \cdot [\vec{\mathbf{r}}'_{l_i} \times \hat{\mathbf{r}}_{r_i}]$$

$$\frac{\partial E}{\partial(\vec{\mathbf{R}} \cdot \hat{\eta})} = -2\sum_i E_i \hat{\mathbf{b}} \cdot \hat{\mathbf{r}}_{\mathbf{r_i}} \times \vec{\mathbf{C}}_{\vec{\mathbf{R}} \cdot \hat{\eta}}(\hat{\mathbf{r}}_{\mathbf{l_i}}), \qquad (10)$$

where $\vec{\mathbf{C}}_{\vec{\mathbf{R}} \cdot \hat{\eta}}(\hat{\mathbf{r}}_{l_i})$ is obtained by differentiating eq. 9. Evaluating the second partial derivatives at the unperturbed solution, where $\hat{\mathbf{b}} = \hat{\mathbf{b}}_\mathbf{e}$ and $\mathbf{b}_T = \mathbf{0}$, gives:

$$\frac{\partial^2 E}{\partial(\mathbf{b}_T \cdot \hat{\epsilon})\partial(\mathbf{b}_T \cdot \hat{\eta})} = 2\sum_i (\hat{\epsilon} \cdot (\vec{\mathbf{r}}'_{l_i} \times \hat{\mathbf{r}}_{r_i}))\hat{\eta} \cdot (\vec{\mathbf{r}}'_{l_i} \times \hat{\mathbf{r}}_{r_i})$$

$$-\hat{\epsilon}\cdot\hat{\eta}(\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}'_{l_i}\times\hat{\mathbf{r}}_{r_i}))^2\Big), \quad (11)$$

$$\frac{\partial^2 E}{\partial(\mathbf{b}_T\cdot\hat{\epsilon})\partial(\vec{\mathbf{R}}\cdot\hat{\eta})} =$$

$$-2\sum_i(\hat{\epsilon}\cdot(\hat{\mathbf{r}}'_{l_i}\times\hat{\mathbf{r}}_{r_i}))(\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}_{r_i}\times\vec{\mathbf{C}}_{\vec{\mathbf{R}}\cdot\hat{\eta}}(\hat{\mathbf{r}}_{l_i})))$$

$$-2\sum_i(\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}'_{l_i}\times\hat{\mathbf{r}}_{r_i}))(\hat{\epsilon}\cdot(\hat{\mathbf{r}}_{r_i}\times\vec{\mathbf{C}}_{\vec{\mathbf{R}}\cdot\hat{\eta}}(\hat{\mathbf{r}}_{l_i}))), \quad (12)$$

and

$$\frac{\partial^2 E}{\partial(\vec{\mathbf{R}}\cdot\hat{\epsilon})\partial(\vec{\mathbf{R}}\cdot\hat{\eta})} =$$

$$2\sum_i[\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}_{r_i}\times\mathbf{C}_{\vec{\mathbf{R}}\cdot\hat{\epsilon}}(\hat{\mathbf{r}}_{l_i}))][\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}_{r_i}\times\vec{\mathbf{C}}_{\vec{\mathbf{R}}\cdot\hat{\eta}}(\hat{\mathbf{r}}_{l_i}))]$$

$$+2\sum_i[\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}_{r_i}\times\hat{\mathbf{r}}'_{l_i})][\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}_{r_i}\times$$

$$\{-4(\hat{\eta}\cdot\hat{\epsilon})\hat{\mathbf{r}}_{l_i}-2\frac{(\hat{\eta}\cdot\hat{\epsilon})}{\sqrt{1-|\vec{\mathbf{R}}|^2}}(\vec{\mathbf{R}}\times\hat{\mathbf{r}}_{l_i})$$

$$-2\frac{(\hat{\epsilon}\cdot\vec{\mathbf{R}})}{\sqrt{1-|\vec{\mathbf{R}}|^2}}(\hat{\eta}\times\hat{\mathbf{r}}_{l_i})-2\frac{(\hat{\epsilon}\cdot\vec{\mathbf{R}})}{(1-|\vec{\mathbf{R}}|^2)^{\frac{3}{2}}}(\hat{\eta}\cdot\vec{\mathbf{R}})(\vec{\mathbf{R}}\times\hat{\mathbf{r}}_{l_i})$$

$$-2\frac{(\hat{\eta}\cdot\vec{\mathbf{R}})}{\sqrt{1-|\vec{\mathbf{R}}|^2}}(\hat{\epsilon}\times\hat{\mathbf{r}}_{l_i})+2(\hat{\eta}\cdot\hat{\mathbf{r}}_{l_i}\hat{\epsilon}+\hat{\epsilon}\cdot\hat{\mathbf{r}}_{l_i}\hat{\eta})\}]. \quad (13)$$

These equations determine the elements of M. M could also be calculated in the spirit of Horn's paper [9], by summing first over all image points prior to calculating the derivatives.

3.2 Determining the N matrix

To determine the N matrix, the partial derivative of the matrix $(\partial E/\partial\mathbf{W})$ with respect to the vector $\mathbf{V}$, the set of image coordinates, should be determined. Define the derivative of the unit ray $\hat{\mathbf{r}}_i$ (either left or right), with respect to its image coordinates $\vec{V}_i$, as the vector function:

$$\mathbf{D}(\hat{\mathbf{r}}_i,\vec{V}_i\cdot\hat{\epsilon})=\frac{\hat{\epsilon}}{\sqrt{f^2+|V_i|^2}}-\frac{(\hat{\epsilon}\cdot\vec{V}_i)}{f^2+|V_i|^2}\hat{\mathbf{r}}_i, \quad (14)$$

where $\hat{\epsilon}$ denotes an arbitrary direction in the image plane, and f is the focal length. Then the derivatives are:

$$\frac{\partial^2 E}{\partial(\mathbf{b}_T\cdot\hat{\epsilon})\partial(V_{l,i}\cdot\hat{\rho})} =$$

$$2[\hat{\epsilon}\cdot(\hat{\mathbf{r}}_{r_i}\times\hat{\mathbf{r}}'_{l_i})][\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}_{r_i}\times R(\mathbf{D}(\hat{\mathbf{r}}_{l_i},V_{l,i}\cdot\hat{\rho})))]$$

$$+2[\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}_{r_i}\times\hat{\mathbf{r}}'_{l_i})][\hat{\epsilon}\cdot(\hat{\mathbf{r}}_{r_i}\times R(\mathbf{D}(\hat{\mathbf{r}}_{l_i},V_{l,i}\cdot\hat{\rho})))], \quad (15)$$

$$\frac{\partial^2 E}{\partial(\mathbf{b}_T\cdot\hat{\epsilon})\partial(V_{r,i}\cdot\hat{\rho})} =$$

$$2[\hat{\epsilon}\cdot(\hat{\mathbf{r}}'_{l_i}\times\mathbf{D}(r_{r,i},V_{r,i}\cdot\hat{\rho}))][\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}'_{l_i}\times\hat{\mathbf{r}}_{r_i})]$$

$$+2[\hat{\epsilon}\cdot(\hat{\mathbf{r}}'_{l_i}\times\hat{\mathbf{r}}_{r_i})][\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}'_{l_i}\times\mathbf{D}(r_{r,i},V_{r,i}\cdot\hat{\rho}))], \quad (16)$$

$$\frac{\partial^2 E}{\partial(\vec{\mathbf{R}}\cdot\hat{\eta})\partial(V_{l,i}\cdot\hat{\rho})} =$$

$$2[\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}_{r_i}\times R(\mathbf{D}(\hat{\mathbf{r}}_{l_i},V_{l,i}\cdot\hat{\rho})))][\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}_{r_i}\times\vec{\mathbf{C}}_{\vec{\mathbf{R}}\cdot\hat{\eta}}(\hat{\mathbf{r}}_{l_i}))]$$

$$+2[\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}_{r_i}\times\hat{\mathbf{r}}'_{l_i})][\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}_{r_i}\times\vec{\mathbf{C}}_{\vec{\mathbf{R}}\cdot\hat{\eta}}(\mathbf{D}(\hat{\mathbf{r}}_{l_i},V_{l,i}\cdot\hat{\rho})))], \quad (17)$$

and

$$\frac{\partial^2 E}{\partial(\vec{\mathbf{R}}\cdot\hat{\eta})\partial(V_{r,i}\cdot\hat{\rho})} =$$

$$2[\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}'_{l_i}\times\mathbf{D}(r_{r,i},V_{r,i}\cdot\hat{\rho}))][\hat{\mathbf{b}}_e\cdot(\vec{\mathbf{C}}_{\vec{\mathbf{R}}\cdot\hat{\eta}}(\hat{\mathbf{r}}_{l_i})\times\hat{\mathbf{r}}_{r_i})]$$

$$+2[\hat{\mathbf{b}}_e\cdot(\hat{\mathbf{r}}'_{l_i}\times\hat{\mathbf{r}}_{r_i})][\hat{\mathbf{b}}_e\cdot(\vec{\mathbf{C}}_{\vec{\mathbf{R}}\cdot\hat{\eta}}(\hat{\mathbf{r}}_{l_i})\times\mathbf{D}(r_{r,i},V_{r,i}\cdot\hat{\rho}))] \quad (18)$$

The above equations determine N. From M and N, the motion error can be calculated as described earlier.

4 Determining the Structure Error

To calculate the estimated structure error due to Horn's algorithm, the matrix of partial derivatives of the structure with respect to the image coordinates is calculated, just as for the motion. The equation for the location of the i-th 3D point, $\mathbf{p}_i$, is:

$$\mathbf{p}_i=\frac{|\mathbf{b}|(\hat{\mathbf{b}}\times\mathbf{R}(\mathbf{I}_{\mathbf{l}_i}))\cdot(\mathbf{R}(\mathbf{I}_{\mathbf{l}_i})\times\mathbf{I}_{\mathbf{r}_i})}{|\mathbf{R}(\mathbf{I}_{\mathbf{l}_i})\times\mathbf{I}_{\mathbf{r}_i}|^2}\mathbf{I}_{r_i}, \quad (19)$$

where the 3D vector $\mathbf{I}=(\mathbf{I}_x,\mathbf{I}_y,f)$ gives the coordinates of the point on the image plane. Recall that $|\mathbf{b}|$, the magnitude of the translation step, is not determined by Horn's algorithm due to the overall scale ambiguity.

$\mathbf{p}_i$ depends on the image coordinates partly through the motion parameters $\mathbf{W}$, which are themselves functions of these coordinates. The partial derivatives of $\mathbf{p}_i$ can therefore be computed by the chain rule. Let the collection of 3D structure estimates be represented by the *state vector* $\mathbf{P}_n=(\mathbf{p}_1,...,\mathbf{p}_m)$, which has length $3m$. Then:

$$d\mathbf{P}_n=\frac{\partial\mathbf{P}_n}{\partial\mathbf{W}}\frac{\partial\mathbf{W}}{\partial\mathbf{V}}d\mathbf{V}+\frac{\partial\mathbf{P}_n}{\partial\mathbf{V}}d\mathbf{V} \quad (20)$$

Since the matrix $(\partial\mathbf{W}/\partial\mathbf{V})$ has already been computed above, only $(\partial\mathbf{P}_n/\partial\mathbf{W})$ and $(\partial\mathbf{P}_n/\partial\mathbf{V})$ need be computed. These are, respectively, $3m\times5$ and $3m\times4m$ matrices, and are given by:

$$\frac{\partial\mathbf{p}_i}{\partial(\mathbf{b}_T\cdot\hat{\epsilon})}=\frac{|\mathbf{b}|(\hat{\epsilon}\times R(\mathbf{I}_{r_i}))\cdot(R(\mathbf{I}_{l_i})\times\mathbf{I}_{r_i})}{|R(\mathbf{I}_{l_i})\times\mathbf{I}_{r_i}|^2}\mathbf{I}_{r_i} \quad (21)$$

$$\frac{\partial\mathbf{p}_i}{\partial(\vec{\mathbf{R}}\cdot\hat{\eta})}=\frac{|\mathbf{b}|(\hat{\mathbf{b}}\times\vec{\mathbf{C}}_{\vec{\mathbf{R}}\cdot\hat{\eta}}(\mathbf{I}_{\mathbf{l}_i}))\cdot(\mathbf{R}(\mathbf{I}_{\mathbf{l}_i})\times\mathbf{I}_{\mathbf{r}_i})}{|\mathbf{R}(\mathbf{I}_{\mathbf{l}_i})\times\mathbf{I}_{\mathbf{r}_i}|^2}\mathbf{I}_{r_i}$$

$$+\frac{|\mathbf{b}|(\hat{\mathbf{b}}\times\mathbf{R}(\mathbf{I}_{\mathbf{l}_i}))\cdot(\vec{\mathbf{C}}_{\vec{\mathbf{R}}\cdot\hat{\eta}}(\mathbf{I}_{\mathbf{l}_i})\times\mathbf{I}_{\mathbf{r}_i})}{|\mathbf{R}(\mathbf{I}_{\mathbf{l}_i})\times\mathbf{I}_{\mathbf{r}_i}|^2}\mathbf{I}_{r_i}$$

$$-2[\frac{(R(\mathbf{I}_{l_i})\times\mathbf{I}_{r_i})\cdot(\vec{\mathbf{C}}_{\vec{\mathbf{R}}\cdot\hat{\eta}}(\mathbf{I}_{l_i})\times\mathbf{I}_{r_i})}{|R(\mathbf{I}_{l_i})\times\mathbf{I}_{r_i}|^2}]$$

$$[\frac{|\mathbf{b}|(\hat{\mathbf{b}}\times\mathbf{R}(\mathbf{I}_{\mathbf{l}_i}))\cdot(\mathbf{R}(\mathbf{I}_{\mathbf{l}_i})\times\mathbf{I}_{\mathbf{r}_i})}{|\mathbf{R}(\mathbf{I}_{\mathbf{l}_i})\times\mathbf{I}_{\mathbf{r}_i}|^2}\mathbf{I}_{r_i} \quad (22)$$

$$\frac{\partial \mathbf{p}_i}{\partial(\mathbf{I}_{l_i} \cdot \hat{\epsilon})} = \frac{|\mathbf{b}|(\hat{\mathbf{b}} \times \mathbf{R}(\hat{\epsilon})) \cdot (\mathbf{R}(\mathbf{I}_{l_i}) \times \mathbf{I}_{r_i})}{|\mathbf{R}(\mathbf{I}_{l_i}) \times \mathbf{I}_{r_i}|^2} \mathbf{I}_{r_i}$$

$$+ \frac{|\mathbf{b}|(\hat{\mathbf{b}} \times \mathbf{R}(\mathbf{I}_{l_i})) \cdot (\mathbf{R}(\hat{\epsilon}) \times \mathbf{I}_{r_i})}{|\mathbf{R}(\mathbf{I}_{l_i}) \times \mathbf{I}_{r_i}|^2} \mathbf{I}_{r_i}$$

$$-2\left[\frac{(R(\mathbf{I}_{l_i}) \times \mathbf{I}_{r_i}) \cdot (R(\hat{\epsilon}) \times \mathbf{I}_{r_i})}{|R(\mathbf{I}_{l_i}) \times \mathbf{I}_{r_i}|^2}\right]$$

$$\left[\frac{|\mathbf{b}|(\hat{\mathbf{b}} \times \mathbf{R}(\mathbf{I}_{l_i})) \cdot (\mathbf{R}(\mathbf{I}_{l_i}) \times \mathbf{I}_{r_i})}{|\mathbf{R}(\mathbf{I}_{l_i}) \times \mathbf{I}_{r_i}|^2} \mathbf{I}_{r_i}\right] \quad (23)$$

and

$$\frac{\partial \mathbf{p}_i}{\partial(\mathbf{I}_{r_i} \cdot \hat{\epsilon})} =$$

$$\frac{|\mathbf{b}|(\hat{\mathbf{b}} \times \mathbf{R}(\mathbf{I}_{l_i})) \cdot (\mathbf{R}(\mathbf{I}_{l_i}) \times \hat{\epsilon})}{|\mathbf{R}(\mathbf{I}_{l_i}) \times \mathbf{I}_{r_i}|^2} \mathbf{I}_{r_i}$$

$$+ \frac{|\mathbf{b}|(\hat{\mathbf{b}} \times \mathbf{R}(\mathbf{I}_{l_i})) \cdot (\mathbf{R}(\mathbf{I}_{l_i}) \times \mathbf{I}_{r_i})}{|\mathbf{R}(\mathbf{I}_{l_i}) \times \mathbf{I}_{r_i}|^2} \hat{\epsilon}$$

$$-2\left[\frac{(R(\mathbf{I}_{l_i}) \times \mathbf{I}_{r_i}) \cdot (R(\mathbf{I}_{l_i}) \times \hat{\epsilon})}{|R(\mathbf{I}_{l_i}) \times R(\mathbf{I}_{r_i})|^2}\right]$$

$$\left[\frac{|\mathbf{b}|(\hat{\mathbf{b}} \times \mathbf{R}(\mathbf{I}_{l_i})) \cdot (\mathbf{R}(\mathbf{I}_{l_i}) \times \mathbf{I}_{r_i})}{|\mathbf{R}(\mathbf{I}_{l_i}) \times \mathbf{I}_{r_i}|^2} \mathbf{I}_{r_i}\right]. \quad (24)$$

Finally, the estimated covariance of the structure can be computed as for the motion.

$$\frac{d\mathbf{P}_n}{d\mathbf{V}} = \frac{\partial \mathbf{P}_n}{\partial \mathbf{W}} \frac{\partial \mathbf{W}}{\partial \mathbf{V}} + \frac{\partial \mathbf{P}_n}{\partial \mathbf{V}}. \quad (25)$$

As before, $Cov(\mathbf{P}_n) = E\{d\mathbf{P}_n\, d\mathbf{P}_n^T\}$, implying that:

$$Cov(\mathbf{P}_n) = \frac{d\mathbf{P}_n}{d\mathbf{V}} Cov(\mathbf{V}) \frac{d\mathbf{P}_n}{d\mathbf{V}}^T = \sigma^2 \frac{d\mathbf{P}_n}{d\mathbf{V}} \frac{d\mathbf{P}_n}{d\mathbf{V}^T}. \quad (26)$$

5 Fusing the New and Old Shape Estimates

The aim is to combine the current structure estimate $\mathbf{P}_n$ with the previous estimate of the shape, $\mathbf{P}_{n-1}^F$, in order to produce a new shape estimate in the current camera frame. We use *shape* rather than *structure* in referring to the combined estimate to emphasise that it is not intriniscally limited in accuracy by the camera pose estimation error, as discussed in section 2. This combination requires that the past estimate be *moved* to the current coordinate frame using the calculated motion parameters. Also, the error in the combined estimate should be updated to this frame.

The moved shape estimate can be written as:

$$\hat{\mathbf{P}}_n = C_n(\mathbf{P}_{n-1}^F) - B_n, \quad (27)$$

where C_n, a $3m \times 3m$ matrix, is given by:

$$C_n = R_n \begin{pmatrix} I_3 & 0 & \dots & 0 \\ 0 & I_3 & \dots & 0 \\ & & \ddots & \\ 0 & 0 & 0 & I_3 \end{pmatrix}. \quad (28)$$

The 3×3 matrix R_n is the estimated rotation between the coordinate frames, and I_3 is the 3×3 identity matrix. Similarly, B_n, a $3m \times 1$ matrix, is given by

$$B_n = \mathbf{b}_n \begin{pmatrix} I_3 \\ I_3 \\ \vdots \\ I_3 \end{pmatrix} \quad (29)$$

where $\mathbf{b}_n$ is the translation between the camera positions. Since the transition matrices C_n and B_n are themselves noisy, this situation differs from the standard recursive measurement process to which the Kalman filter is normally applied.

The expected error of $\hat{\mathbf{P}}_n$ in the current coordinate system is again computed by linearising:

$$d\hat{\mathbf{P}}_n = \frac{\partial \hat{\mathbf{P}}_n}{\partial \mathbf{P}_{n-1}^F} d\mathbf{P}_{n-1}^F + \frac{\partial \hat{\mathbf{P}}_n}{\partial \mathbf{W_n}} \frac{\partial \mathbf{W_n}}{\partial \mathbf{V_n}} d\mathbf{V_n}. \quad (30)$$

Define the $3m \times 5$ matrix A_n by:

$$A_n = \left[\frac{\partial \hat{\mathbf{P}}_n}{\partial \mathbf{W}_n}\right] \left[\frac{\partial \mathbf{W_n}}{\partial \mathbf{V}_n}\right]. \quad (31)$$

Then the covariance is:

$$Cov(d\hat{\mathbf{P}}_n d\hat{\mathbf{P}}_n^T) = C_n Cov(d\mathbf{P}_{n-1}^F d\mathbf{P}_{n-1}^{FT}) C_n^T + A_n Cov(d\mathbf{V}_n d\mathbf{V}_n^T) A_n^T. \quad (32)$$

Here it has been assumed that $\mathbf{V}_n$ and $\mathbf{P}_{n-1}^F$ are statistically independent; most likely, after a few frames, the correlation between the left image and the previous combined estimate is small and the independence assumption is valid.

Finally, to determine the new estimate $\mathbf{P}_n^F$, the result of Horn's algorithm and $\hat{\mathbf{P}}_n$ must be combined. This is straightforward except for the overall scale ambiguity in recovering motion and structure. Ideally, the scale should be removed from the state vector; currently, the scale of the translation step is fixed to be its exact value as obtained from ground truth. Then, the standard Kalman filter result [7] for the fused estimate is:

$$\mathbf{P}_n^F = Cov(\mathbf{P}_n^F)[Cov(\hat{\mathbf{P}}_n)^{-1} \hat{\mathbf{P}}_n + Cov(\mathbf{P}_n)^{-1} \mathbf{P}_n], \quad (33)$$

where $Cov(\mathbf{P}_n^F)$ is the covariance of the combined estimate at this camera position:

$$Cov(\mathbf{P}_n^F) = [Cov(\hat{\mathbf{P}}_n)^{-1} + Cov(\mathbf{P}_n)^{-1}]^{-1} \quad (34)$$

$Cov(\mathbf{P}_n)$, the estimated error of Horn's algorithm in determining structure, has already been calculated in Section 4. Thus, all the ingredients have been assembled for computing the fused shape.

6 Experiments

A series of experiments has been carried out using the algorithm described above for one synthetic and two real image sequences. To measure how accurately shape has been determined, a procedure is employed that eliminates the effects of error in the camera pose estimate. We use Horn's absolute orientation algorithm [10] to map the shape estimate into the coordinate system of the ground truth. The measure of shape error is the average distance between the true and estimated positions of the 3D points after this registration. Absolute orientation is also carried out for Horn's two frame reconstruction of the structure, for purposes of comparison with the present algorithm.

The parameters of our experimental sequences are shown in Table 1. In the first experiment, a synthetic image sequence was used. 20 points were tracked over 13 image frames, with depths ranging from 43 to 65 units. In a fixed coordinate system, the camera translated towards the scene in a sig–sag path around the fixed z–direction. The translation step varied in magnitude from 0.44 to 0.6, and the rotations were about the vertical (y) direction, and had magnitude 1.5°. In the camera coordinate system, the translation was always in the current z–direction. The image noise was Gaussian, with $\sigma = 0.3$ pixels, modelling a flow error $\sigma = 0.6$.

The average 3D error for the recursive algorithm, displayed in Table 2.1, decreases swiftly and essentially monotonically, contrasting with the 2 frame error shown in the Table. These results were achieved even though the translational motion was largely into z, with a very small baseline to triangulate with. If one outlier point is excluded, the average structure error for our algorithm improves significantly; after the last iteration, it is only 1.7 units. Table 2.2 displays the depths recovered by the recursive algorithm after the last frame for all points *prior* to using absolute orientation. Since for this sequence the motion is relatively well determined, the depth estimates for the estimated pose correspond fairly well to the ground truth. The sizes of the transformations computed by absolute orientation are shown in Table 2.3. Compared to the two–frame results, a much smaller transformation is required to register the recursive structure estimate with the ground truth. Note that the recursive results do not seem to be strongly affected by the large error produced by the two–frame measurement in iteration 8.

The second experiment employs an image sequence previously discussed in [15]. The first image of this sequence is displayed in Figure 1. The motion consisted of a box rotating around its approximately vertical body axis in steps of about 3.6°, with a stationary camera. The average 3D error for the recursive algorithm decreases dramatically compared to the two–frame result after just 3–4 iterations (Table 3.1). This fast decrease can be attributed to correctly combining successive measurements to obtain an effectively wider baseline. Note that the errors in the final iterations are comparable to the accuracy of the ground truth, estimated to be about 1.5 mm. This accuracy is high considering the large depths ($\sim$ 600 mm).

In Table 3.2, the depths *after registration* are displayed for the last frame. There are no outliers. Finally, the sizes of the transformations for absolute orientation are shown in Table 3.3.

For the third experiment, the rocket–field image sequence from the IEEE motion workshop database was used. This is an outdoor sequence produced using a camera mounted on an autonomous vehicle; the first image is displayed in Figure 2. Ground truth was available for 12 of the 15 points tracked over 11 frames [4]. Points were mostly tracked by hand but some were obtained by running Anandan's optical flow algorithm [1]. The robot motion was a constant forward translation (in steps of about 0.9 m) and very small rotations.

In Table 4.1 the average 3D distance error for 10 of the 12 tracked points is displayed without registration; absolute orientation could not be performed since there was no valid closed form solution. One of the 2 points excluded was too far away (almost four times as far as the next most distant point) and the other was highly erroneous as it remained near the FOE. For comparison, the results of simply averaging the two–frame structures are also shown. The averaging was done recursively by using the 2-frame motion estimate to move the previous average into the current estimated camera frame, where it was then averaged with the new structure estimate. The percentage errors in depth are displayed in Table 4.2.

Though the two–frame error fluctuates, the recursive error falls rapidly and consistently, in contrast with the results reported in [3] for a recursive algorithm which does not include the effects of motion error. After ten frames, it is better by a factor of two than the result of blind averaging. This is so despite the fact that this is not a favorable case for the recursive algorithm, since the translational motion into the scene gives similar rather than complementary structure estimates from each frame pair, as compared, for example, with the box sequence.

References

[1] P. Anandan, *Measuring Visual Motion from Image Sequences,* PhD Thesis, COINS Tech. Report TR 87-21, Univ. Of Mass. at Amherst, MA., 1987.

[2] T. J. Broida and R. Chellappa, "Estimating the Kinematics and Structure of a Rigid Object from a Sequence of Monocular Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 13, no. 6, pp. 497–513, 1991.

[3] N. Cui, J. Weng and P. Cohen, "Extended Structure and Motion Analysis from Monocular Image Sequences," *Proceedings 3rd IEEE International Conference on Computer Vision,* Osaka, Japan, 1990, pp. 222-229.

[4] R.Dutta, R.Manmatha and L.R.Williams, "A Data Set for Quantitative Motion Analysis," *CVPR,* San Diego, California, pp. 159-164, 1989.

[5] R. Dutta and M. Snyder, "Robustnesss of Correspondence-Based Structure from Motion," *Proceedings 3rd IEEE International Conference on Computer Vision*, Osaka, Japan, Dec. 1990.

[6] O. D. Faugeras, N. Ayache, and B. Faverjon, "Building Visual Maps by Combining Noisy Stereo Measurements," *IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp. 1433-1438, 1986.

[7] Technical Staff, The Analytical Sciences Corp., and A. Gelb, ed., *Applied Optimal Estimation*, MIT Press, 1986.

[8] J. Heel, 'Dynamic Motion Vision," *Image Understanding Workshop*, Palo Alto, CA, pp. 702-713, 1989.

[9] B. K. P. Horn, "Relative Orientation," *International Journal of Computer Vision*, Vol. 4, pp. 59-78, 1990.

[10] B. K. P. Horn, "Closed Form Solution of Absolute Orientation Using Unit Quaternions," *J. Opt. Soc. Am.* A, vol. 4, pp. 629-642, 1987.

[11] R. V. R. Kumar, A. Tirmalai and R.C. Jain, "A Nonlinear Optimisation Algorithm for the Estimation of Structure and Motion Parameters," *CVPR*, San Diego, CA, pp. 136-143, 1989.

[12] L. Matthies, R. Szeliski, and T. Kanade, "Incremental Estimation of Dense Depth Maps from Image Sequences," *CVPR*, Ann Arbor, Michigan, pp. 366-374, 1988.

[13] L. Matthies, T. Kanade, and R. Szeliski, "Kalman Filter-Based Algorithms for Estimating Depth from Image Sequences," *International Journal of Computer Vision*, vol 3, pp. 209-236, 1989.

[14] J. Oliensis and J. I. Thomas, "Incorporating Motion Error in Multi-frame Structure from Motion," *Proceedings IEEE Workshop on Visual Motion*, Princeton, pp 8-13, 1991.

[15] H. S. Sawhney, J. Oliensis, and A. R. Hanson, "Description and Reconstruction from Image Trajectories of Rotational Motion", in *ICCV*, Osaka, Japan, December, 1990, pp. 494-498.

[16] M. Spetsakis and J. Aloimonos, "A Multi-frame Approach to Visual Motion Perception," *Proc. IEEE Workshop on Motion*, Irvine, CA, March, 1989.

[17] J. Inigo Thomas and J. Oliensis, "Fusing Structure by Kalman Filtering," TR 90-93, COINS, UMASS, May 1990.

[18] J. Inigo Thomas and J. Oliensis, "Incorporating Motion Error in Multiframe Structure from Motion," *7th Scandinavian Conference on Image Analysis*, Denmark, pp. 950-957, 1991.

[19] C. Tomasi and T. Kanade, "Shape and Motion without Depth", *IUW*, Pittsburgh, PA, pp. 258-270, 1990.

Sequence	# Frames	# pts.	Depth Range	FOV	Focal length	Image
Synthetic	13	20	43–65	45°	16 mm	512 × 512
Box	8	12	585–707 mm	(23.4°,22.4°)	16 mm	256 × 242
Rocket-Field	11	15	18–53 m	(71.9°, 56.8°)	6 mm	512 × 512

Table 1. Parameters for the three motion sequences used.

Iter.	1	2	3	4	5	6	7	8	9	10	11
2 Frame	22.2	12.0	15.2	25.9	10.2	17.9	14.5	24.0	10.3	14.8	11.9
Recur.	11.5	11.9	9.9	9.3	8.6	7.6	5.9	5.0	4.5	3.6	3.4

Table 2.1. Average 3D error after absolute orientation for synthetic sequence. For the recursive algorithm, most of the error in 9–11 is due to a single outlier point.

True	37.0	43.2	59.4	46.8	59.5	56.9	39.8	44.9	46.9	38.0
2 Frame	37.3	37.4	61.1	48.9	72.1	61.0	31.0	46.8	38.8	40.1
Recursive	36.7	44.4	56.5	45.4	59.5	54.1	39.4	43.1	46.6	37.7
True	52.9	41.9	39.2	45.7	44.7	41.8	46.9	55.0	56.8	55.8
2 Frame	43.3	34.5	44.1	38.2	14.9	22.7	89.6	61.6	77.3	60.2
Recursive	52.9	42.0	39.2	45.6	30.3	38.4	46.4	55.2	56.1	50.2

Table 2.2. Unregistered depths in last frame for 20 tracked points.

2	Rot. (dg)	-10	-8	-22	-4	-23	-19	-19	-9	-6	-13
F	Trans.	13.7	18.6	42.3	6.7	27.9	18.3	35.8	8.9	17.9	13.9
R	Scale	0.87	0.73	0.32	0.85	0.46	0.95	0.42	0.99	0.77	0.81
R	Rot. (dg)	-6	-1	-0.2	-0.5	-0.8	-0.5	-0.9	-0.8	-1.2	-1.0
E	Trans.	11.8	8.7	8.0	6.7	5.4	3.8	3.0	2.4	1.9	1.8
C	Scale	0.91	0.94	0.95	0.96	0.96	0.98	0.99	0.99	1.00	1.00

Table 2.3. Magnitudes of the rotation, translation, scale used for registration in the last 10 iterations.

Iteration	1	2	3	4	5	6	7
2 Frame (mm)	4.66	4.88	13.97	3.01	5.55	6.37	5.87
Recursive (mm)	4.67	10.9	3.02	1.15	0.88	1.51	1.7

Table 3.1. Average 3D error after absolute orientation for box sequence.

True	598.7	611.9	589.1	600.6	654.9	629.9	644.8	665.8	669.7	687.8	629.7	616.2
2 Fr.	607.0	613.9	595.4	596.2	649.0	635.2	648.3	668.6	657.8	685.2	631.2	611.3
Rec.	600.3	611.7	591.0	600.5	652.8	630.4	644.0	665.0	667.9	687.1	632.6	615.8

Table 3.2. Depths (mm) *after registration* in last frame for 12 tracked points.

2 Fr. Rot. (dg)	-2.0	-3.9	-12.4	-2.2	-2.2	-3.2	-2.1
Recur. Rot. (deg)	-2.0	-12.3	-1.9	-0.2	-0.3	-1.1	-2.1
2 Fr. Trans. (mm)	21.79	46.76	160.20	25.08	24.07	44.27	22.89
Recur. Trans. (mm)	21.98	152.97	26.64	3.64	3.55	14.09	24.48
2 Fr. Scale	1.07	1.15	.54	1.10	0.94	0.84	1.14
Recur. Scale	1.08	1.60	0.94	1.05	1.06	0.98	1.10

Table 3.3. Magnitudes of the registration transformations.

Iteration	1	2	3	4	5	6	7	8	9	10
2 Frame (m)	11.9	6.2	18.8	7.9	3.2	5.3	9.6	12.5	10.5	10.1
Blind Avg (m)	11.9	7.0	10.2	9.4	8.0	7.1	6.9	7.0	7.1	6.4
Recursive (m)	11.9	5.6	6.4	7.0	5.3	3.3	3.0	2.6	2.5	2.4

Table 4.1. Average 3D error for 10 points in rocket-field sequence.

Iteration	1	2	3	4	5	6	7	8	9	10
2 Frame	26.1	12.1	28.5	14.8	10.2	15.8	21.1	27.9	25.6	21.3
Blind Avg	26.1	14.8	17.3	14.8	13.5	14.1	12.6	13.7	13.7	12.0
Recursive	26.1	9.9	12.0	12.5	10.4	8.2	8.2	7.1	6.4	5.8

Table 4.2. Average percentage error in depths for 10 points.

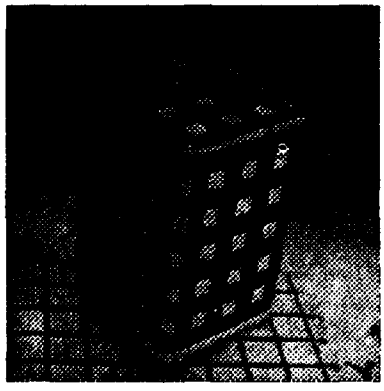


Figure 1. Box sequence



Figure 2. Rocket-field sequence

A Computational Model for Binocular Motion Processing

Poornima Balasubramanyam and M. A. Snyder

Department of Computer Science
University of Massachusetts at Amherst *

Abstract

Let a binocular camera system move through the environment. It is possible to determine a three dimensional field of vectors, where each vector is parallel to the actual inducing 3D velocity of an imaged point (relative to the moving camera) scaled by the depth of that point. This representation enables a more realistic description of the relative 3D velocity between the environment and the camera than is afforded by considering the information in either the optic flow field or the disparity field alone. This vector field (termed the *p-field*) may then be employed in the processing of a sequence of binocular images to determine the camera motion parameters and depth of the viewed scene. In this paper, we present this p-field, and discuss its computation from the image measurable quantities: optic flow and stereo disparity.

1 Introduction

The computation of depth is important for many aspects of vision. One popular source of depth information is stereo imagery. Another source is time-varying imagery from which the motion of the environment relative to the camera system is determined and from this depth is computed. Our goal is the integrated processing of these two sources of depth information.

One of the earliest uses of the term *stereoscopic motion* was in [Regan et al., 79]. They provide evidence to support models of neural organisations in the human visual system that are binocularly triggered by changing disparity for the purpose of detecting motion in depth. The computational model developed here has been strongly motivated by such psychophysical evidence.

There have been many differing attempts to integrate stereo and motion processing. Some, such as the work of [Jenkin, 84] and [Waxman et al., 86], have concentrated on addressing the early (and similar) correspondence issues. Jenkin used both stereo and motion information in a prediction–correction formulation, while Waxman et al. established a correlation between the stereo disparity and the relative flow between the stereo pair. Other work has primarily addressed the problem of relative motion and depth recovery. Mitiche [Mitiche, 88] used depth from stereo with information from the optic flow to impose rigidity constraints on the actual 3D motion. Current approaches [Matthies, 88, Zhang et al., 88] typically employ the stereo module as the main source of depth information. This is used to guide the motion analysis over a sequence of images, as well as to plan the robot motions that would simplify the stereo correspondence problem. Often approaches are embedded in a Kalman Filter framework in order to bound the errors over a sequence of stereo frame-pairs.

The model of binocular motion processing that we propose in this paper is motivated by a need to develop a *single framework* for binocular motion processing within which we can then address each of these issues.

1.1 Our Approach

In this work, we consider the *dynamics* of a scene as viewed by a stereo pair of cameras. This subsumes both the analysis of static stereo imagery at one time instant to obtain the static disparity between the two images and thereby depth, and the analysis of a monocular motion pair to obtain the optic flow for a pair of frames and thereby relative motion and depth. We are interested in utilising binocular motion in order to obtain integrated additional constraints that can then be employed to determine the motion and depth without depending on a single (and possibly erroneous) source for the depth or on mere juxtaposition of the results from independent processing of the motion and stereo information.

The model presented here depends on a relation between the ratio of the relative optic flow in a binocular pair of images and the disparity, derived originally in [Waxman et al., 86]. Employing this result, we demonstrate that it is possible to derive the vector parallel to the real instantaneous 3D velocity vector scaled by the depth of the point, located at the image of the 3D point. This vector is derived using purely image measurable quantities, i.e., the optic flow and disparity. This field of scaled 3D vectors is referred to as the *p-field*, a term originally coined in [Scott, 86], although not in the

*This research was supported by DARPA under TACOM contract DAAE07-91-C-R035

context of binocular motion. The *p*-field has significant implications about the phenomenon of binocular motion since it implies that at *the lowest levels of visual processing, where traditionally only the 2D entities located on the image plane were considered, it is now possible to examine and exploit the nature of 3D phenomena directly.*

We are currently interested in the use of the *p*-field as a framework within which to represent the problems of flow/disparity computation, the computation of the 3D motion as well as independent object motion. For instance, observing that the *p*-vector is the scaled real 3D velocity vector, it is shown [Balasubramanyam, 92] to be more appropriate to impose smoothness constraints (spatial and temporal) on this vector field than on the optic flow field. This was examined by [Scott, 86] in the context of spatial smoothing, but not within the framework of binocular motion. It was also not recognised that observing the changing environment with a binocular system enables us to actually compute this 3D field.

We are also interested in the interpretation of available flow and disparity information for the estimation of the motion parameters and depth within this domain. For instance, in the case of ideal pure translation, the *p*-field directly yields the direction of translation. Since the assumption of ideal translation is not very realistic in most experimental situations [Dutta et al., 88], our current work is focused on examining the effects that small rotations have on such an assumption of pure translation. In the case of general motion, several possible algorithms for the computation of the motion parameters are being examined.

2 The Mathematical Model

In this section, we briefly describe a model for combining information from motion and stereo when the binocular sensor is in motion in the environment. More details may be found in [Balasubramanyam, 91]. It is shown that the temporal changes in the two images of the stereo camera system have a very specific relationship to each other, and to the changing environment. An instantaneous description of the relative motion between the camera system and the environment is employed in order to establish the relationship between the various image measurable entities available from binocular imagery. The measured image entities are the two separate optic flow fields for the left and right image pairs, and the stereo disparity at the first time instance.

Consider a camera system with a fixed right hand cartesian coordinate reference, (X^c, Y^c, Z^c). Without any loss of generality, we can assume that the optical axis coincides with the Z^c axis, and that the focal length is normalised to unity, i.e., all distances are measured in units of "focal length". The center of projection is at the camera origin, and the image is formed on the image plane $\mathcal{I}$, $Z^c = 1$. Let the camera system have instantaneous motion relative to the environment described by a translational velocity, $\mathbf{t} = (T_X, T_Y, T_Z)$, and an angular velocity, $\omega = (\omega_X, \omega_Y, \omega_Z)$, about an axis passing through the origin of the coordinate reference frame attached to the camera system.

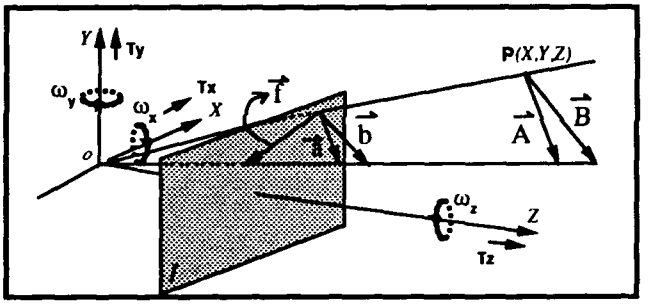


Figure 1: Let $\vec{A}, \vec{B}$ be two possible instantaneous 3D velocity vectors of point P such that they project to the *same* 2D image velocity vector $\vec{f}$. The resulting *p*-vectors are $\vec{a}, \vec{b}$ respectively, disambiguating between the two inducing 3D instantaneous velocity vectors.

The motion of a stationary environmental point $\mathbf{P} = (X, Y, Z)$, in the coordinate system attached to the camera, is given by :

$$\dot{\mathbf{P}} = -(\omega \times \mathbf{P} + \mathbf{t}). \tag{1}$$

The *p-vector*, $\mathbf{p}(x, y)$ is defined [Scott, 86] as

$$\mathbf{p} = \frac{\dot{\mathbf{P}}}{Z}, \tag{2}$$

where Z is the depth of the point $\mathbf{P}$. This is the vector *parallel to the instantaneous 3D velocity vector of the environmental point relative to the camera* and scaled inversely by the depth of the point. We denote the set of vectors, $\Pi = \{\mathbf{p}(x, y); (x, y) \in \mathcal{D}\}$, where $\mathcal{D}$ denotes the image domain (assume a discrete grid of points (x, y)), as the *p-vector field*. Now, under 3D rotations of the coordinate system, the *p*-vector does not transform like a vector. This rotational *non-covariant* property implies that the *p*-vector is not really a vector; nevertheless the terminology has achieved some legitimacy, and is preserved here.

Assuming perspective projection, the components of the *p*-vector are:

$$\mathbf{p} \stackrel{\text{def}}{=} \frac{1}{Z}\begin{pmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \begin{pmatrix} \omega_Z y - \omega_Y - \frac{T_X}{Z} \\ \omega_X - \omega_Z x - \frac{T_Y}{Z} \\ \omega_Y x - \omega_X y - \frac{T_Z}{Z} \end{pmatrix} \tag{3}$$

This *p*-vector is considered with reference to the image of the 3D point of interest on the image plane (x, y), and has components in the image-plane (p_x, p_y) *as well as* a component orthogonal to the image plane, p_z (see Figure 1).

The image velocity field (approximated in computation by the optic flow field) is denoted as $\Phi = \{(u(x, y), v(x, y)); (x, y) \in \mathcal{D}\}$, where $\mathcal{D}$ denotes the image domain as before.

Consider a parallel-axes binocular camera system (i.e., no vergence). The translational displacement of

the two axes is described by a horizontal baseline of $2b_x$, and a vertical baseline of $2b_y$. We assume $f \equiv 1$ for both cameras. A (virtual) cyclopean coordinate reference frame is placed in the middle, between the left and right camera coordinate reference frames. Superscripts l, r, c denote the entities with respect to the left, right and cyclopean references, respectively. With respect to the cyclopean origin, the displacement of the left reference origin, O^l is given by $\mathbf{b^l} = (b_x, b_y, 0)^T$, and the displacement of the right reference origin, O^r, is given by $\mathbf{b^r} = (-b_x, -b_y, 0)^T$.

A point in the environment projects to (x^l, y^l) in the left image and to (x^r, y^r) in the right image. The *disparity field* for environmental points between their corresponding left and right images is denoted by $\Delta = \{\vec{\delta}(x,y); (x,y) \in \mathcal{D}\} = \{(\delta_x(x,y), \delta_y(x,y)); (x,y) \in \mathcal{D}\}$, where the image domain $\mathcal{D}$ can be considered synonymous with the left, right or cyclopean image planes. In the following, the cyclopean image domain is chosen as the reference domain. The disparity vector is then given by the stereo triangulation equations:

$$\vec{\Delta} = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} x^l - x^r &=& \frac{-2b_x}{Z^c} \\ y^l - y^r &=& \frac{-2b_y}{Z^c} \end{pmatrix}. \qquad (4)$$

2.1 The p-vector

Assume that the binocular system has the geometry described above, and let the cyclopean coordinate reference frame have motion described by an angular velocity $\omega^c = (\omega^c_X, \omega^c_Y, \omega^c_Z)$ about an axis passing through the cyclopean origin, and a translational velocity $\mathbf{t^c} = (T^c_X, T^c_Y, T^c_Z)$, with respect to the stationary environment. The instantaneous velocity of a point $\mathbf{P} = (X^c, Y^c, Z^c)$ can be given, as before, by:

$$\dot{\mathbf{P}}^{\mathbf{c}} = -(\omega^c \times \mathbf{P^c} + \mathbf{t^c}). \qquad (5)$$

Denoting the image velocity fields for the left and right images by $\dot{\Phi}^l(x^c, y^c)$ and $\dot{\Phi}^r(x^c, y^c)$ respectively, the difference of the image velocities in the two images, $\dot{\Phi}^r(x^c, y^c) - \dot{\Phi}^l(x^c, y^c)$ can then be shown to be (see [Balasubramanyam, 91] for details):

$$\begin{aligned} \frac{u^r - u^l}{\delta_x} &= \frac{v^r - v^l}{\delta_y} \\ &= -y^c\omega^c_X + x^c\omega^c_Y - \frac{T^c_Z}{Z^c}. \end{aligned} \qquad (6)$$

Comparing this relative flow with Equation 3, we see that

$$\frac{u^r - u^l}{\delta_x} = \frac{v^r - v^l}{\delta_y} = p_z, \qquad (7)$$

Here, p_z is the component of the p-vector that describes the scaled relative 3D velocity along the optical axis (orthogonal to the image plane).

In [Waxman et al., 86], the binocular camera geometry modelled consists of only relative horizontal displacement between the camera origins (i.e., $b_y = 0$), and Equation 6 is termed the *relative flow*. This result is employed in [Balasubramanyam, 87] to recover the motion in depth parameters, $(\omega^c_X, \omega^c_Y, T^c_Z)$. It can be seen that accomodating a vertical displacement in the camera geometry, and thus accounting for a vertical disparity, merely provides an additional source of information about the relative flow.

Thus, given the image measurable quantities, (u^l, v^l), (u^r, v^r), and (δ_x, δ_y) for an image point $(x^l, y^l) \Leftrightarrow (x^r, y^r)$, we obtain p_z, which in turn can be used with the flow to compute the remaining components of the p-vector with respect to the left and right image coordinates as $[\mathbf{p^l} = (p^l_x, p^l_y, p_z)^T, \mathbf{p^r} = (p^r_x, p^r_y, p_z)^T]$, where the components are given as:

$$\left.\begin{aligned} p^j_x &= u^j + x^j p_z \\ p^j_y &= v^j + y^j p_z \end{aligned}\right\} \text{ for } j = l, r. \qquad (8)$$

By symmetry, we determine the cyclopean p-vector as:

$$\mathbf{p^c} = \begin{bmatrix} p^c_x \\ p^c_y \\ p_z \end{bmatrix} = \begin{bmatrix} \frac{u^l+u^r}{2} + \frac{x^l+x^r}{2} p_z \\ \frac{v^l+v^r}{2} + \frac{y^l+y^r}{2} p_z \\ \frac{u^r-u^l}{\delta_x} = \frac{v^r-v^l}{\delta_y} \end{bmatrix} \qquad (9)$$

The p-vector yields an elegant natural description of the inducing instantaneous velocity when viewed binocularly. If the camera geometry is assumed to have no vertical (Y) displacement between the two camera origins, then the *relative vertical flow* between the two images becomes indeterminate, implying that the computation of p_z depends entirely on the relative horizontal flow component. In general, in the limit as the baseline between the two cameras tends to zero, and the geometry collapses to a monocular motion situation, the relative flow components (both horizontal and vertical) become indeterminate. Thus, the orthogonal p_z component (and thereby the other two components, p_x, and p_y) cannot be determined. In a similar vein, as the baseline between the cameras decreases, the error in determining the p_z component will increase dramatically, exhibiting unstable behavior due to small values for both the relative flow components and the stereo disparity. A large baseline, relative to the depth of interest, is important for robust p-field computation.

[Balasubramanyam, 91] contains a discussion of the properties exhibited by the p-field for pure rotational as well as pure translational motion. This paper also briefly discusses an algorithm for determining the motion parameters in the case of general motion, including preliminary results on real data.

In ongoing work, the p-field model is being developed for a binocular viewing geometry accomodating vergence between the two camera axes. This model is consistent (as is to be expected) with the development here in that it demonstrates that the relative motion between the binocular image planes over time yields the (scaled) component of the 3D motion along the optical axis, and that this can be further employed to describe the complete (scaled) 3D motion.

3 Uncertainty Analysis

It is seen from equation 9 that optic flow estimates from the left and right image pairs as well as stereo disparity estimates are required to compute the p-field. Given that these match estimates are susceptible to noise in real image scenarios, it is important to understand the dependence of the uncertainty in the p-vector estimate on the uncertainty in the flow and disparity estimates. The second order statistics of the p-vector estimate are crucial to all algorithms that use the p-field for the recovery of motion, depth and motion boundaries. We briefly present the covariance analysis for the computed p-vector. More details may be found in [Balasubramanyam, 92].

The required initial optic flow estimates are obtained from the hierarchical correlation algorithm described in [Anandan, 89]. A version of the same algorithm is modified to search along a narrow window centered on the epipolar constraint line in order to obtain the initial disparity estimates.

The (3×3) covariance matrix associated with the computed p-vector is obtained by propagating the (2X2) covariance matrices associated with the flow and disparity estimates. We assume that a specific matching algorithm [Anandan, 89] is employed to obtain the flow and disparity estimates and the associated covariances. However, the analysis applies equally to any algorithm as long as it returns the displacement estimates with their 2×2 covariance matrices or provides a way of determining these matrices.

In the matching algorithm described in [Anandan, 89], the Hessian of the sum-of-squared-differences (SSD) error surface is used to obtain the confidence weights and confidence directions associated with the displacement match estimate. If $\mathbf{H}$ is the Hessian of the SSD error surface, and σ_n^2 is the variance of uncorrelated Gaussian noise, then it is shown in [Szeliski 88] that for a given match displacement estimate, $\mathbf{d}$, the associated 2×2 covariance matrix $\Sigma_{\mathbf{dd}}$ is given by:

$$\Sigma_{\mathbf{dd}} = 2\sigma_n^2 \mathbf{H}^{-1}. \qquad (10)$$

Thus, with each flow and disparity match estimate $\mathbf{d}$, the 2×2 covariance matrix $\Sigma_{\mathbf{dd}}$ is known (from equation 10). Let $\mathbf{J_{px}}$ denote the Jacobian that defines the functional dependence of the random variable $\mathbf{p} = (p_x, p_y, p_z)^T$ on the random variable $\mathbf{x} = (u^l, v^l, u^r, v^r, \delta_x)^T$. This may be derived employing equation 9. $\mathbf{\Sigma_{xx}}$ denotes the covariance matrix for the variable $\mathbf{x}$, and may be obtained by applying equation 10 to each of the flow and disparity match estimates, assuming that these estimates are obtained independently. Thus, the covariances for the match estimates can be propagated to obtain the covariance matrix for the p-vector estimate, $\mathbf{\Sigma_{pp}}$, as:

$$\mathbf{\Sigma_{pp}} = \mathbf{J_{px}}^T \, \mathbf{\Sigma_{xx}} \, \mathbf{J_{px}}. \qquad (11)$$

In general, the covariance matrix for the p-vector $\mathbf{\Sigma_{pp}}$ is not a diagonal matrix since the elements are correlated. It can be shown [Balasubramanyam, 92] that this covariance matrix can be used to establish a confidence ellipsoid around the estimated p-vector value, where the semi-axes of the ellipsoid are determined by diagonalizing $\mathbf{\Sigma_{pp}}$.

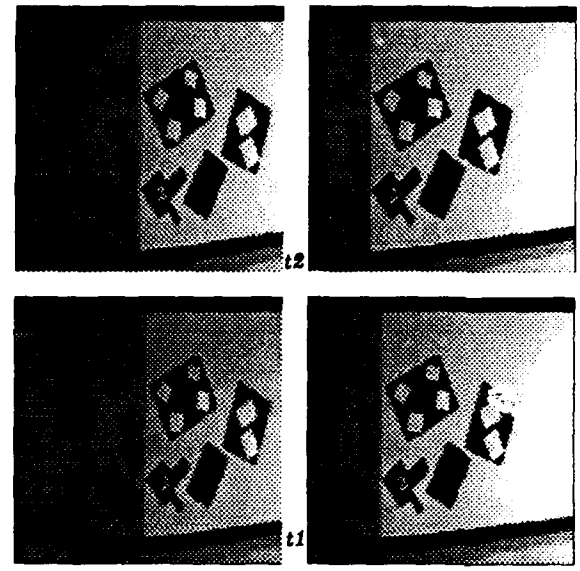


Figure 2: The two stereo image pairs viewed by a binocular camera system which is undergoing nearly translational motion at time instances *t1* and *t2*.

4 Experimental Results

The grey-scale stereo images at two time instances are part of a data set from an indoor run with the "Harvey" robot vehicle at UMass. The binocular parallel axes of the camera pair had a baseline of 20 inches. The distance from the vehicle center to the corner point of intersection of the two walls with floor (Figure 2) was approximately 18 feet. The motion of the vehicle was toward the corner. Absolute ground truth for the camera motion is not known. The motion is a largely translational motion directed along the optical axis of the camera pair. The hierarchical correlation algorithm [Anandan, 89] is employed to obtain the flow and disparity estimates for image resolution of 128 X 128. No smoothing is performed on the optic flow or disparity estimates that are used to compute the p-field (shown in Figure 3). It can be observed that since the motion of the camera is *toward* the viewed scene, the p-vectors for most of the pixels are directed correctly parallel to the optical axis (since p_x and p_y components ≈ 0). Significant errors occur at points in the left portion of the left image where considerable occlusion occurs between the stereo pair as well as between the right motion pair. Other sources of significant error are horizontal lines in the four intensity images since they lead to erroneous horizontal flow and stereo disparity estimates, and thereby erroneous p_z components. Covariance analysis on this p-field yields estimates for the uncertainty in the computed p-vectors. This is employed as a filter on the computed p-field and the resulting filtered field is shown in Figure 4. This filtering step

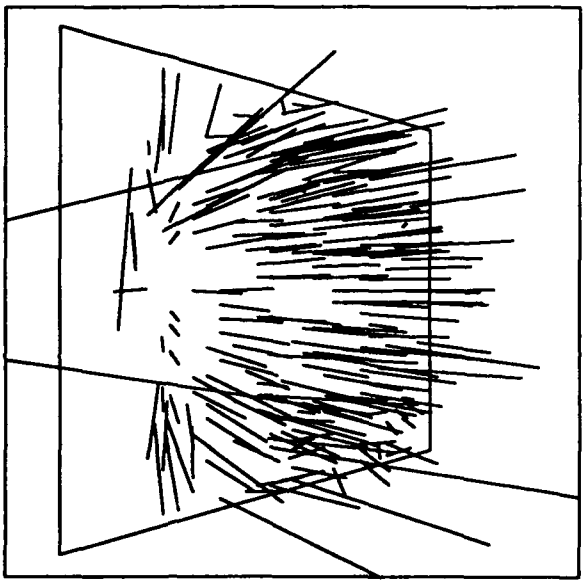


Figure 3: The p-vectors, shown at every fourth image row and column location, computed from unsmoothed optic flow and disparity estimates.

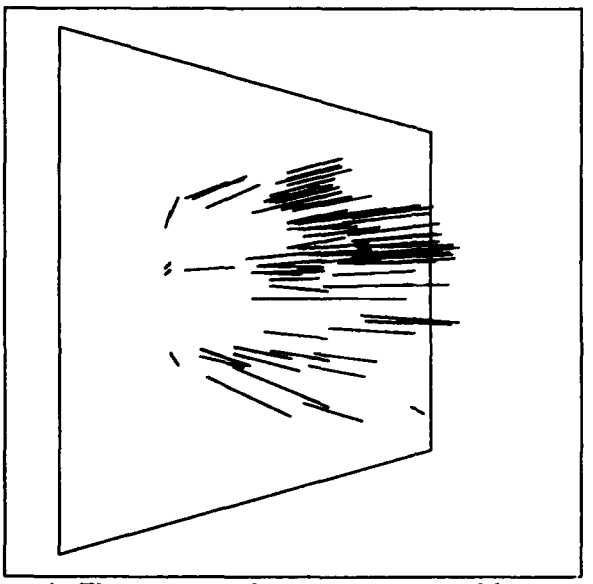


Figure 4: The p-vectors, shown at every second image row and column location, with uncertainty estimates used to filter out significant errors.

is adopted here only to demonstrate that the covariance analysis enables the elimination of many vectors with errors. In current work we are looking at spatial and temporal smoothing of the p-fields computed over multiple stereo sequences using the uncertainty estimates to weight the individual vectors.

5 Conclusion

The p-field is proposed as a model for binocular motion processing. It provides a natural description of the 3D motion of a camera system relative to the environment since it retains the directional motion information. Hence, the p-field can be used to model smooth 3D motion. Rigidity constraints can be imposed more naturally on such a 3D field and we can examine the problem of independent object motion detection and occlusion detection from this context as well. We present the p-field as a framework to examine issues in binocular motion processing.

References

[Anandan, 89] P. Anandan, "A Computational Framework and Algorithm for the Measurement of Visual Motion," *IJCV*, Vol. 2, 1989, 283-310.

[Balasubramanyam, 87] P. Balasubramanyam, "Computation Of Motion-in-Depth Parameters Using Stereoscopic Motion Constraints " *IEEE Computer Society Workshop on Computer Vision*, Miami, 1987, 349-351.

[Balasubramanyam, 91] P. Balasubramanyam and M. A. Snyder, "The P-Field: A Computational Model for Binocular Motion Processing," *Proc. IEEE CVPR*, Hawaii, 1991, 115-120.

[Balasubramanyam, 92] P. Balasubramanyam, *PhD Dissertation*, University of Massachussetts at Amherst, 1992.

[Dutta et al., 88] R. Dutta, R. Manmatha, E. Riseman and M. A. Snyder, "Issues in Extracting Motion Parameters and Depth from Approximate Translational Motion," *COINS TR 88-52*, University Of Massachussetts, Amherst, 1988.

[Jenkin, 84] M. R. M. Jenkin, "The Stereopsis of Time-Varying Images," *Technical Report*, University of Toronto, Toronto, Ontario, Canada, (RBCV-TR-84-3), Sept. 1984.

[Matthies, 88] L. H. Matthies, "Depth from Stereo Image Sequences," *International Workshop on High Precision Navigation*, Stuttgart, Germany, May 1988.

[Mitiche, 88] A. Mitiche, "A Computational Approach to the Fusion of Stereo and Kineopsis," *Motion Understanding: Robot and Human Vision, edt. by W. N. Martin and J. K. Aggarwal*, 1988, 81-95.

[Regan et al., 79] D. Regan and K. I. Beverley, "Binocular and Monocular Stimuli for Motion-In-Depth: Changing Disparity and Changing Size Inputs Feed the Same Motion-In-Depth Stage," *Vision Research*, 1979, 19, 1331-1342.

[Scott, 86] G. L. Scott, "Smoothing the Optic Flow under Perpective Projection," *Proc. IEEE CVPR*, 1986, 504-509.

[Sseliski 88] R. Sseliski, "Bayesian Modeling of Uncertainty in Low Level Vision," *PhD Dissertation*, CMU, Aug. 1988.

[Waxman et al., 86] A. M. Waxman and J. H. Duncan, "Binocular Image Flows: Steps toward stereo-motion fusion," *IEEE PAMI*, Vol. 8, Nov. 1986, 715-729.

[Zhang et al., 88] Z. Zhang, O. Faugeras and N. Ayache,"Analysis of a Sequence of Stereo Scenes Containing Multiple Moving Objects Using Rigidity Constraints,"*Proc. 2nd International Conference on Computer Vision* 1988, Florida.

Behavioral Visual Motion Analysis

Yiannis Aloimonos, Zoran Duriç, Cornelia Fermüller,
Liuqing Huang, Ehud Rivlin, and **Rajeev Sharma**
Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3411

Abstract

We propose here a new approach to addressing problems related to visual motion, namely the purposive approach [4]. Instead of considering the various visual motion tasks as applications of the general structure from motion module, we consider them as independent problems and we directly seek solutions for them. As a result we can achieve unique and robust solutions without having to compute optic flow and without requiring a full reconstruction of the visual space, because it is not needed for the tasks. In the course of the exposition, we present novel solutions to various important visual tasks related to motion, such as the problems of motion detection by a moving observer, passive navigation, relative-depth computation, 3-D motion estimation, and visual interception, using as input only the spatial and temporal derivatives of the image intensity function. It turns out that the spatiotemporal derivatives of the image (i.e. the so-called normal flow) do not seem to be capable of solving the general "structure from motion" problem. They are, however, sufficient to provide robust algorithms for the solution of many interesting visual tasks that do not require the full solution, but only part of it. The ability to create robust nontrivial behaviors suggests the possibility that visual perception could be studied as intelligent behavior. We point out some of the benefits and drawbacks of this paradigm that studies vision as a set of behaviors that recover the visible world partially, but well enough to carry out a task (purposive, animate or behavioral vision), and we contrast it to the traditional paradigm of treating vision as a general recovery problem.

1 Introduction and Motivation

The problem of structure from motion has attracted a lot of attention in the past several years [23, 30, 33] because of the general usefulness that a potential solution to this problem would have. Important navigational problems, such as detection of independently moving objects by a moving observer, passive navigation, obstacle detection, target pursuit, and many other problems related to robotics, teleconferencing, etc., would be simple applications of a structure from motion module. The problem has been formulated as follows: Given a sequence of images taken by a monocular observer (the observer and/or parts of the scene could be moving), to recover the shapes (and relative depths) of the objects in the scene, as well as the (relative) 3-D motions of independently moving bodies.

The problem has been formulated and usually treated as an aspect of the general task of recovering 3-D information from motion [25, 19]. The majority of the proposed solutions to date are based on the following modular and hierarchical approach:

1. First, one computes the optic flow on the image plane, i.e. the velocity with which every image point appears to be moving.[1]
2. Then segmentation of the flow field is performed and different moving objects are identified on the image plane. From the segmented optic flow one then computes the 3-D motion with which each visible surface is moving relative to the observer. (Assuming that an object moves rigidly, a monocular observer can only compute its direction of translation and its rotation, but not its speed.)
3. Finally, using the values of the optic flow along with the results of the previous step, one computes the surface normal at each point, or equivalently, the ratio Z_i/Z_j of the depths of any two points i and j.

The reason that most approaches have followed the above three-step approach is two-fold. The first is due to the formulation of the problem, which insists on recovering a complete relative depth map and accurate three-dimensional motion. The second is due to the fact that the constraints relating retinal motion to three-dimensional structure involve 3-D motion in a nonlinear manner that does not allow separability. For examples of such approaches, see [1, 34, 23, 30]. However, the past work in this paradigm, despite its mathematical elegance, is far from being useful in real-time navigational systems, and such techniques have found few or no prac-

[1] For clarity, we consider only the differential case. In the case of long range motion one computes discrete displacements, but the analysis remains essentially the same.

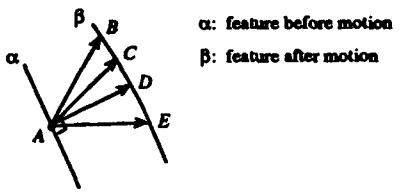


Figure 1: The aperture problem. Point A could have moved to B, C, D, E. However, whatever the value of the image motion vector is, its projection on the normal to α is always AD (known).

tical applications.[2] Consequently, this approach cannot be used yet to explain the ability of biological organisms to handle visual motion.

There exist many reasons for the limitations of the optic flow approach, related to all three steps listed above. To begin, the computation of optic flow is an ill-posed problem, i.e. unless we impose additional constraints, we cannot estimate it [19]. Such constraints, however, impose a relationship on the values of the flow field which is translated into an assumption about the scene in view (for example, smooth). Thus, even if we are capable of obtaining an algorithm that computes optic flow in a robust manner, the algorithm will work only for a restricted set of scenes. The only available constraint at every point (x, y) of the changing image $I(x, y, t)$ for the flow (u, v) is the constraint $I_x u + I_y v + I_t = 0$ [21], where the subscripts denote partial differentiation. This means that we can only compute the projection of the flow on the gradient direction $((I_x, I_y) \cdot (u, v) = -I_t)$, i.e. the so-called *normal flow*. More graphically, it means that if a feature (for example, an edge segment) in the image moves to a new position, we don't know where every point of the segment moved to (see Figure 1); we only know the normal flow, i.e. the projection of the flow on the image gradient at that point.

A second reason has to do with the very essence of optic flow. An optic flow field is the vector field of apparent velocities that are associated with the variation of brightness on the image plane. Clearly, the scene is not involved in this definition. One would hope that optic flow would be equivalent to the so-called motion field [19], which is the (perspective) projection on the image plane of the three-dimensional velocity field associated with each point of the visible surfaces in the scene. However, the optic flow field and the motion field are not equal in general. Verri and Poggio [36] reported some general results in an attempt to quantify the difference between the optic flow and motion fields. Although we don't yet have necessary and sufficient conditions for the equality of the two fields, it is clear that they are equal only under specific sets of restrictive conditions.

A third reason is related to the second step of the existing algorithms for structure from motion. These algorithms attempt to first recover three-dimensional motion before they proceed to recover relative depth, and this problem of 3-D motion appears to be very sensitive in the presence of small amounts of noise in the input (flow or displacements) [31, 38, 1, 2].

In [31] several experiments as well as comparisons with various algorithms were made and the finding was that an average error of 1% to 2% in the input (retinal correspondence) can create an error of about 100% in the estimated parameters. An important question to ask then is what makes this problem unstable, and to seek ways to address any inherent instabilities that might arise. There is recent work towards this direction but difficult questions still remain.

But while theoretical research on the principles of structure from motion continues in its quest for optimal recovery, we can also follow an alternative approach. We can ask the following simple question: if we had a robust structure from motion module, what would we use it for? The answer of course lies in a taxonomy of visual tasks involving motion, i.e. navigational tasks. A few such *generic* navigational tasks are, for example, the following:

- *Detection of independently moving objects in the environment, by a moving observer.* This is a nontrivial task, as everything moves on an image obtained by a moving observer, thus making it hard to distinguish independent motion. Although many general schemes have been proposed for segmentation of a flow field into areas corresponding to differently moving objects, there are still problems in practical applications involving more than one independently moving object. Other approaches of interest are those that combine measurements of flow with some 3-D interpretation which can then be used for incremental improvements to segmentation in an iterative manner. However, no practical robust system for detecting independently moving objects in general environments and based on optic flow has been demonstrated to date.
- *Passive navigation.* Passive navigation is a term used to describe the processes by which a system can determine its motion with respect to the environment. This is important for kinetic stabilization which, in its simplest form, requires a system to maintain a fixed position and attitude in space in the presence of perturbing influences. More generally, stabilization can refer to any conditions placed on the motion parameters; for instance, the system might be required to translate without rotation. The two abilities are interrelated because stabilization is generally achieved by bringing the motion parameters to certain specified values. The capacity for passive navigation is prerequisite for any other navigational ability. In order to guide the system, some idea of the present motion and some method of setting it to known values must be available. In present robot systems the necessary information is often explicitly available as a result of a built-in coordinate system. For an autonomously moving system, however, there must be an active sensing capacity. It is possible to obtain the required in-

[2] Possible exceptions are photogrammetry and semiautonomous applications requiring a human operator.

formation mechanically as is done by the inertial guidance systems in guided missiles. However, the task can also be performed by visual means and it is this problem that we address here.

- *Obstacle avoidance.* Obstacle avoidance refers, simply, to the ability to utilize sensory information to maneuver in an environment containing physical objects without striking them. This can be considered a second-level ability. It requires some capacity for passive navigation, but little else, and could thus be considered the lowest level of active navigation. This task can be performed non-visually by range sensing methods, and it has been generally proposed that the problem be solved visually with a similar algorithm utilizing depth data from a scene reconstructed by the structure from motion module.
- *Avoidance of collision with a moving object.* A robust structure from motion module can detect the 3-D motion of a moving object, calculate its 3-D position with the aid of a binocular system and predict its three-dimensional trajectory. Thus, it can detect any possibilities for collision, by reconstructing the 3-D trajectory of the moving object.
- *Understanding of relative depth.* Visual motion provides a very rich amount of information about the relative depths of objects in the environment (which object is closer). Clearly, this is one of the outputs of the structure from motion module.
- *Visual pursuit.* A three dimensional visual pursuit system consists of an eye (camera(s)), a subject, an object and a mind. The mind uses information from the eye in order to control the movement of the subject so that it will collide with (intercept, catch) the object. Under the traditional paradigm of considering vision as a recovery problem, visual pursuit is just another application of the structure from motion module. In such a case, the camera would reconstruct the three dimensional positions and motions of the camera, the subject and the object and then this information would be utilized by a planning module to generate correct control of the subject.

Given the lack of success in developing a robust structure from motion module, it would seem reasonable to consider simpler problems. There are visual problems, such as the above, which do not require the full realization of the structure from motion capability, yet which are both nontrivial and possess the sort of environmental invariance that would give them general utility. To consider a few examples from biological navigation, the housefly can maneuver visually in three dimensions in a complex environment without striking obstacles; a number of bees and wasps can recognize and return to a particular location in their environment; and the frog can extend its tongue and catch flying insects. Human beings can also perform such tasks, but obviously they can be performed with far less computational equipment than humans possess. We propose here to consider, in the context of navigational tasks, some of the above problems, more specific and more restricted than the general structure from motion problem, with a view towards producing examples of visual systems that have the potential for robustness. This approach is termed purposive [4].

We show later that specific questions such as the ones above can be answered without having to go through the estimation of optic flow. The derivatives of the image intensity function are enough for the task. The approach taken in this paper calls for the solution of specific visual tasks, such as the ones above, in such a way that the solution does not have more power than it is supposed to have. For example, the procedure that provides relative depth is designed only for that purpose and cannot be used to solve the passive navigation problem, or the problem of 3-D motion estimation. Of course, if information about 3-D motion is known, it can be effectively utilized in the estimation of relative depth, but this is of no concern to us here. When building a system that can deal with visual motion problems, we can visualize it as consisting of many processes working in a cooperative manner to solve various problems. For example, the theories described in this paper could be used to design a process that computes relative depth from image measurements, independently of the process that computes 3-D motion. However, after a number of computational steps, when results about relative depth and 3-D motion become available from the two independent processes, they can be exchanged and the constraints relating them can be effectively utilized so that the results are as consistent as possible.

2 Qualitative Methods

Most visual navigation tasks, including the ones described above, have been considered to be subproblems for the reconstructive school. The connection is a natural one since most of these tasks involve shape and distance relationships between the system and the environment which can be expressed in terms of the quantitative idiom of the reconstructive school. This perception has tended to discourage explicit research on such specific problems by classifying them as special cases of an important general problem. It has also tended to obscure the fact that many of the operations necessary to implement specific visual tasks can be expressed in qualitative terms which are more aptly described in terms of the recognition idiom. Consider, for example, the problem of passive navigation. It is not necessary to know exactly how the system is moving with respect to the environment but only whether it is rotating or translating at all, and if so, in what direction forces must be applied to reduce the motion. In the case of obstacle avoidance, the most relevant information is not the exact distance in centimeters from the observer to each point in the environment, but whether the observer is on a collision course with a nearby obstacle and if so, in which direction it should move to avoid the danger of a crash. The common factor in these examples is that they do not require precise quantitative information and that in each case, the information necessary to carry out the task can be represented in a space having only a few degrees of freedom.

3 Organization of the Paper

We wish to develop the mathematics that will give rise to general solutions to the specific problems of detection of independent motion, passive navigation, relative depth estimation, obstacle avoidance, estimating whether an object is on a collision course with the observer, and visual pursuit using the derivatives of the image as input, as opposed to considering them as applications of the structure from motion module. Section 4 is devoted to the description of the input and Sections 5–9 describe general solutions to the specific tasks mentioned above. Finally, Section 10 is devoted to the presentation of some experimental results. It should be pointed out that here we are mostly interested in the theoretical principles behind these perceptual processes and the geometry of the normal flow. We seek solutions that have uniqueness properties using normal flow as input, since normal flow is well defined, while optic flow is not. Thus, we only present the computational theory behind each process. For various properties of the solutions of the individual problems, a theoretical error analysis and an extensive implementation, see [16, 18, 22, 29]. It will become clear that solving the abovementioned problems using normal flow (which contains less information than optic flow) becomes possible only through the employment of an active visual agent [5]. The reason is, of course, that some of the computational burden is transferred to the activity of the agent.

4 The Input

Our motivation is by now clear. We wish to avoid using optic flow as the input to visual motion tasks. On the other hand, we must utilize some description of the image motion. As such a description we choose the spatial and temporal derivatives $\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial t}$ of the image intensity function $I(x, y, t)$. These quantities define the normal flow at every point, i.e. the projection of the optic flow on the direction of the gradient (I_x, I_y). Clearly, estimating the normal flow is much easier than estimating the actual optic flow. But how is normal flow related to the three-dimensional motion field? Is the normal optic flow field equal to the normal motion field, and under what conditions? This question was first investigated by Verri and Poggio [36]

Let $I(x, y, t)$ denote the image intensity, and consider the optic flow field $\vec{v} = (u, v)$ and the motion field $\vec{\bar{v}} = (\bar{u}, \bar{v})$ at a point (x, y) where the local (normalized) intensity gradient is $\vec{n} = (I_x, I_y)/\sqrt{I_x^2 + I_y^2}$. The normal motion field at point (x, y) is by definition

$$\begin{aligned} \bar{u}_n &= \vec{\bar{v}} \cdot \vec{n} && \text{or} \\ \bar{u}_n &= \frac{(I_x, I_y)}{\sqrt{I_x^2 + I_y^2}} \cdot \left(\frac{dx}{dt}, \frac{dy}{dt}\right) && \text{or} \\ \bar{u}_n &= \frac{\nabla I}{\|\nabla I\|} \cdot \left(\frac{dx}{dt}, \frac{dy}{dt}\right) && \text{or} \\ \bar{u}_n &= \frac{1}{\|\nabla I\|}\left(I_x \frac{dx}{dt} + I_y \frac{dy}{dt}\right) \end{aligned}$$

Similarly, the normal optic flow [21] is

$$u_n = -\frac{1}{\nabla I} I_t$$

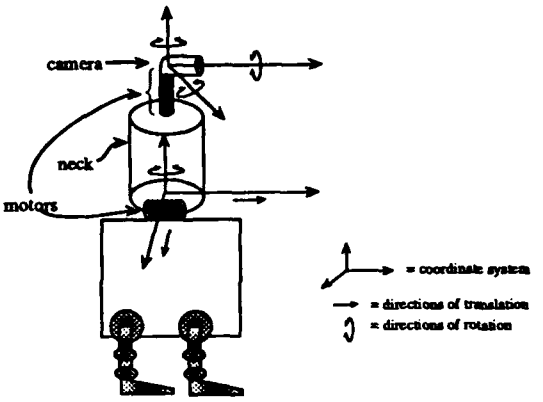


Figure 2: The active observer.

Thus $$\bar{u}_n - u_n = \frac{1}{\nabla I} \frac{dI}{dt}$$

From this equation it follows that if the change of intensity of an image patch during its motion $(\frac{dI}{dt})$ is small enough (which is a reasonable assumption) and the local intensity gradient has a high magnitude, then the normal optic flow and motion fields are approximately equal. Thus, provided that we measure normal flow in regions of high local intensity gradients, the normal flow measurements can safely be used for inferring 3-D structure and motion.

We are now ready to describe our solution to the various motion related tasks. Since the input to the perceptual process is the normal flow, and the normal flow field contains less information than the motion field, in order to solve various problems we need to transfer much of the computation to the activity of the observer [5]. A geometric model of the observer is given in Figure 2. Notice that the camera is resting on a platform ("neck") with six degrees of freedom (actually only one of the degrees is used) and the camera can rotate around its x and y axes (saccades).

5 Passive Navigation

5.1 A qualitative solution

The problem of passive navigation (kinetic stabilization) has attracted a lot of attention in the past ten years [13, 23, 24, 34, 31, 33] because of the generality of a potential solution. The problem has been formulated as follows: Given a sequence of images taken by a monocular observer undergoing unrestricted rigid motion in a stationary environment, to recover the 3-D motion of the observer. In particular, if (U, V, W) and (A, B, C) are the translation and rotation, respectively, comprising the general rigid motion of the observer, the problem is to recover the following five numbers: the direction of translation $(\frac{U}{W}, \frac{V}{W})$ and the rotation (A, B, C) (see Figure 3). The problem has thus been formulated as the general 3-D motion estimation problem (kinetic depth or structure from motion) and its solution would solve several other problems.

Consider a model for a monocular observer as in Figure 3. We assume that the observer moves forward. It

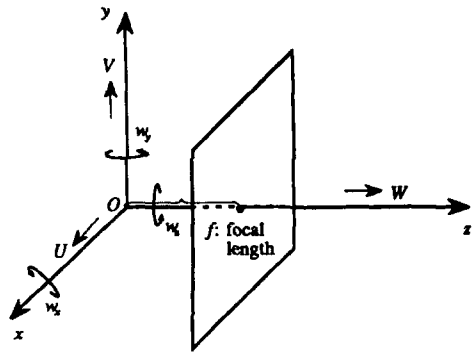


Figure 3: Geometric model of the observer.

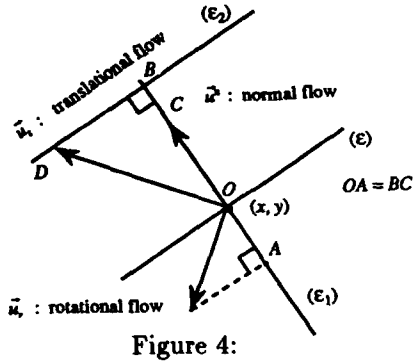


Figure 4:

should be noted that the observer is equipped with inertial sensors which provide the rotation (A, B, C) of the observer at any time. As the observer moves in its environment, normal flow fields are computed in real time. Since optic flow due to rotation does not depend on depth but on image position (x, y), we know (and can compute in real time) its value (u^R, v^R) at every image point along with the normal flow.[3] That means that we know the geometrical locus of the optic flow due to translation (see Figure 4). Since the observer moves forward in a static scene, it is approaching anything in the scene and the flow is expanding. From Figure 4, it is clear that the focus of expansion (FOE) $(\frac{U}{W}, \frac{V}{W})$ (when the gradient space of directions $(\frac{U}{W}, \frac{V}{W})$ is superimposed with the image space) lies in the half plane defined by line ϵ. Clearly, at every point we obtain a constraint line which constrains the FOE to lie in a half plane. If the FOE lies on the image plane (i.e. the direction of translation is anywhere inside the sector $OABCD$ (Figure 5)) then the FOE is constrained to lie in an area on the image plane and thus it can be localized (see Figure 6). When the FOE does not lie inside the image, a closed area cannot be found, but the votes collected by the half planes indicate its general direction. In this case the observer, with a "saccade" (a rotation of the camera), can bring the FOE inside the image and localize it (Figure 7 explains the process).

[3]If computation of normal flow at some points is unreliable, we just don't compute normal flow there.

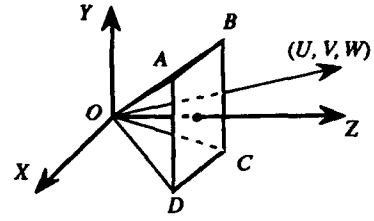
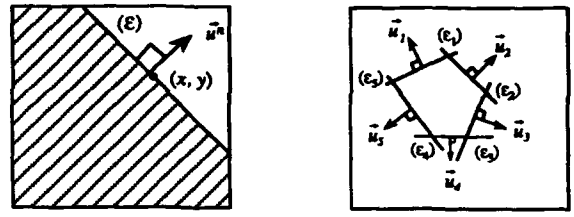


Figure 5: Consider the camera coordinate system. If the translation vector (U, V, W) is anywhere inside the solid $OABCD$ defined by the nodal point of the eye and the boundaries of the image, then the FOE is somewhere on the image.



(a) (b)

Figure 6: (a) From a measurement $\vec{u}$ of the normal flow due to translation at a point (x, y) of the image, every point of the image belonging to the half plane defined by ϵ that does not contain $\vec{u}$ is a candidate for the position of the focus of expansion, and collects one vote. The voting is done in parallel for every image measurement. (b) If the FOE lies within the image boundaries, then the area containing the highest number of votes is the area containing the FOE. Using only a few measurements can result in a large area. Using many measurements (all possible) results in a small area (in our experiments an area of at most three or four pixels).

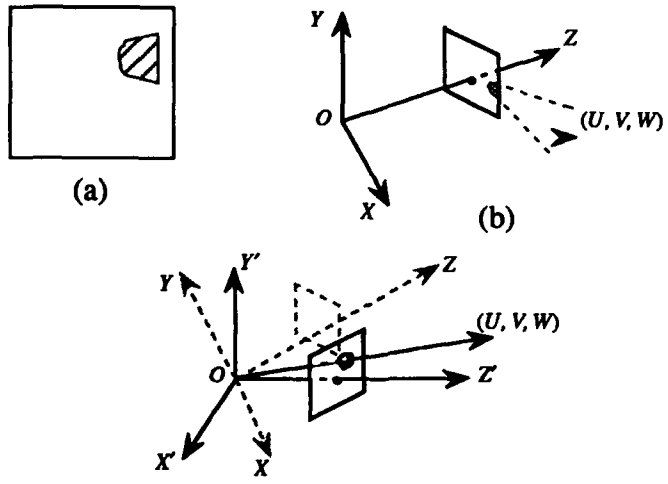


Figure 7: (a) If the area containing the highest number of votes has a piece of the image boundary as part of its boundary, then the FOE is outside the image plane (b). (b) The position of the area containing the highest number of votes indicates the general direction in which the translation vector lies. (c) The camera ("eye") rotates so that the area containing the highest number of votes becomes centered. With a rotation around the x and y axes only, the optical axis can be positioned anywhere in space. The process stops when the highest vote area is entirely inside the image.

5.2 The algorithm

We assume that the computation of the normal flow, the voting and the localization of the area containing the highest number of votes can happen in real time. In this paper we don't get involved with real time implementation issues as we wish to analyze the theoretical aspects of the technique. However, it is quite clear that computation of normal flow can happen in real time (there already exist chips performing edge detection). According to the literature on Hough transforms and connectionist networks [9], voting could also happen in real time. Let S denote the area with the highest number of votes. Let $L(S)$ be a Boolean function that is true when the intersection of S with the image boundary is the null set, and false otherwise. Then the following algorithm finds area S. We assume that the inertial sensors provide the rotation and thus we know the normal flow due to translation.

```
1.  begin {
2.          find area S
3.          repeat until L(S)
4.          { rotate camera around x, y axes
              so that the optical axis passes
              through the center of S (saccade)
5.          find area S
              }
            output S
            }
```

If the camera has a wide angle lens, then image points can represent many orientations, and only one saccade may be necessary. But if we have a small angle lens, then we may have to make more than one saccade.[4]

5.3 Improvement of the solution

It is clear that the technique just described provides as an answer an area on the image containing the FOE. How large or small this area can be depends on the distribution of surface markings and thus on the measured normal flow. If the FOE lies in a featureless area, the resulting area will not be small. For some applications the knowledge of area S might be enough to accomplish the task. We can, however, narrow down a more accurate solution, with S providing one constraint.

Assuming that inertial sensors provide us with the rotation, we can derotate the normal flow field. Thus, assuming a translational normal flow field $v_n(x, y)$, we have: $v_n = u \cdot n_x + v n_y$, where (u, v) is the optic flow and (n_x, n_y) the direction of the gradient at that point. Since we have derotated, the optic flow is

$$u = \frac{U - xW}{V}, v = \frac{V - yW}{Z}$$

and thus

$$v_n = -\frac{W}{Z}(xn_x + yn_y) + \frac{W}{Z}\left(n_x\frac{U}{W} + n_y\frac{V}{W}\right)$$

or

$$\frac{n_x}{v_n}\frac{U}{W} + \frac{n_y}{u_n}\frac{V}{W} - \frac{Z}{W} = \frac{n_x}{v_n}\cdot x + \frac{n_y}{u_n}\cdot y$$

This is a linear equation in the FOE $\left(\frac{U}{W}, \frac{V}{W}\right)$ and the time to collision with every scene point.

If we consider a small image patch P with Z_{av} the average depth of the scene points giving rise to the patch under consideration, then the above equation, for every point $(x_i, y_i) \in P$ with depth z_i, can be written as

$$\frac{n_x}{v_n}\frac{V}{W} + \frac{n_y}{v_n}\frac{V}{W} - \frac{Z_{\text{av}}}{W} = \frac{n_x}{v_n}x + \frac{n_y}{v_n}y + \left(\frac{Z_i}{W} - \frac{Z_{\text{av}}}{W}\right)$$

The expected value of the last term in the above equation is zero, and assuming that we can correctly compute (n_x, n_y) and v_n, equations

$$\frac{n_x}{v_n}\frac{U}{W} + \frac{n_y}{v_n}\frac{V}{W} - \frac{Z_{\text{av}}}{W} = \frac{n_x}{v_n}x + \frac{n_y}{v_n}y$$

at every point $(x, y) \in P$ constitute a linear system in the unknowns $\frac{U}{W}$, $\frac{V}{W}$ and $\frac{Z_{\text{av}}}{W}$. Solving such systems for several patches robustly provides the FOE and a hazard map (showing different time-to-collision values). The patches need at least three normal flow measurements, and so they can be quite small.

5.4 Analysis of the method

We have assumed that the inertial sensors will provide the observer with accurate information about rotation. Although expensive accelerometers can achieve very high accuracy, the same is not true for inexpensive inertial sensors and so we are bound to have some error. Thus we must assume that some unknown rotational part still exists and contributes to the value of the

[4] Up to this point the algorithm is similar to [20]. However, as will become clear later, it works even when rotation is present, while in [20] the solution works only for translational motion.

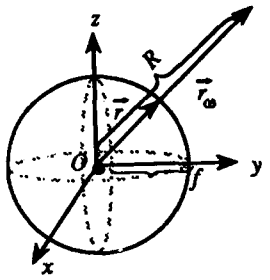


Figure 8:

normal flow. As a result, the method for finding the FOE (previous section) which is based on translational normal flow information (since we have "derotated") might be affected by the presence of some rotational flow. In this section, we study the effect of rotation (the error of the inertial sensor) on the technique for finding the FOE. At the same time *we provide a technique for bounding the FOE given a normal flow field containing both rotation and translation.*

In order to avoid artificial problems introduced by perspective distortions in the case of a planar retina and to simplify the formulas without loss of generality, we employ a spherical retina. Let a sphere with radius f and center O (Figure 8) represent the spherical retina (with O the nodal point of the eye) and a coordinate system $OXYZ$ attached to it. Let

$$\vec{r}_\omega = (X, Y, Z) \text{ be a world point}$$

and $\vec{r} = (x, y, z)$ be its image on the image plane.

Then $$\frac{\vec{r}}{f} = \frac{\vec{r}_\omega}{R},\ R = \|\vec{r}_\omega\| = \sqrt{\vec{r}_\omega \cdot \vec{r}_\omega}$$

In the sequel we derive expressions for optic (normal) flow in the new configuration.

If the velocity of the world point $\vec{r}_\omega$ is given by

$$\dot{\vec{r}}_\omega = -\vec{t} - \vec{\omega} \times \vec{r}_\omega$$

where $\vec{t}$ is translation $(\vec{t} = (U, V, W))$
$\vec{\omega}$ is rotation $(\vec{\omega} = (\omega_x, \omega_y, \omega_z))$

$$\text{then } \dot{\frac{\vec{r}}{f}} = \frac{\dot{\vec{r}}_\omega \cdot R - \vec{r}_\omega \cdot \dot{R}}{R^2}$$

$$\dot{R} = \frac{d}{dt}\left(\sqrt{\vec{r}_\omega \cdot \vec{r}_\omega}\right) = \frac{1}{2R}(\dot{\vec{r}}_\omega \cdot \vec{r}_\omega + \vec{r}_\omega \cdot \dot{\vec{r}}_\omega) = \frac{\dot{\vec{r}}_\omega \cdot \vec{r}_\omega}{R}$$

We have

$$\begin{aligned}\dot{\frac{\vec{r}}{f}} &= \frac{\dot{\vec{r}}_\omega}{R} - \frac{\vec{r}_\omega}{R^3}(\dot{\vec{r}}_\omega \cdot \vec{r}_\omega) \\ &= -\frac{\vec{t}}{R} - \frac{\vec{\omega} \times \vec{r}_\omega}{R} - \frac{\vec{r}_\omega}{R} \cdot \frac{1}{R^2}\left((-\vec{t} - \vec{\omega} \times \vec{r}_\omega) \cdot \vec{r}_\omega\right) \\ &= -\frac{\vec{t}}{R} - \frac{\vec{\omega} \times \vec{r}}{f} + \frac{\vec{r}}{f} \cdot \frac{1}{R}(\vec{t} \cdot \frac{\vec{r}}{f})\end{aligned}$$

or

$$\begin{aligned}\dot{\vec{r}} &= -\frac{\vec{t}f}{R} - \vec{\omega} \times \vec{r} + \frac{\vec{r}}{Rf}(\vec{t} \cdot \vec{r}) \qquad (1) \\ &= \frac{1}{R}\left[-\vec{t}f + \frac{\vec{r}}{f}(\vec{t} \cdot \vec{r})\right] - \vec{\omega} \times \vec{r}\end{aligned}$$

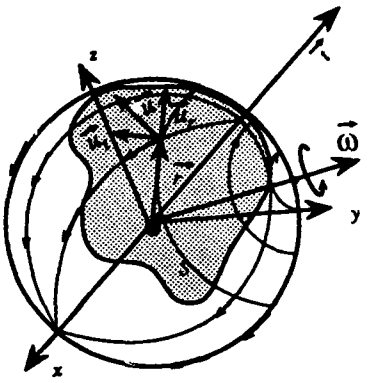


Figure 9:

Thus, the translational flow is

$$\vec{u}_t = \frac{1}{R}\left[-\vec{t}f + \frac{\vec{r}(\vec{t} \cdot \vec{r})}{f}\right]$$

while the rotational flow is given by

$$\vec{u}_R = -\vec{\omega} \times \vec{r}$$

Without loss of generality we can set $f = 1$.

At this point we define two quantities that will be of use later. They are $\tau = \frac{R}{\|\vec{t}\|}$, which we term time to collision, and $k = \frac{\|\vec{\omega}\|}{\|\vec{t}\|}R = \|\vec{\omega}\|\tau$, which represents the effective ratio of rotation and translation.

The geometry of the spherical projection is then given in Figure 9. It has been shown [28] that a full (360°) visual field simplifies motion analysis. However, what we usually have is just a piece of the surface of the sphere (due to a limited field of view). Assume then that the image (the part that we see) is projected on the surface patch S. Obviously, voting for the estimation of the FOE can be performed for all points on S.

5.4.1 Principles of voting

Consider

$\vec{r}_i = (x, y, z)$, a point in S,
$\vec{n}_i = (n_x, n_y, n_z)$, the image gradient direction at point $\vec{r}_i$,
$\dot{\vec{r}}_i = \vec{u}_i = (u_x, u_y, u_z)$, the flow at point $\vec{r}_i$, and
$\vec{u}_i^n = (\vec{n}_i \cdot \vec{u}_i) \cdot \vec{n}_i$, the normal flow at $\vec{r}_i$.

Then (see Figure 10) if $\vec{r} = (x, y, z)$ is a point in S, a feature point $\vec{r}_i$ will vote for $\vec{r}$ being the FOE (direction of translation) iff $\vec{u}_i^n(\vec{r} - \vec{r}_i) < 0$ (see Figure 10).

If $V[\vec{r}]$ represents the number of votes collected at point $\vec{r}$, then it is easy to see that

$$V[\vec{r}] = \sum_{\vec{r}_i \in S} U\left[\vec{u}_i^n(\vec{r}_i - \vec{r})\right]$$

where $U(x) = \begin{cases} 1, & x > 0 \\ 0, & x \le 0 \end{cases}$ (Heaviside function)

Let $S' = \left\{\vec{r} | \forall \vec{r}' \in S,\ V[\vec{r}] \ge V[\vec{r}']\right\}$ be the set of points that have acquired the maximum number of votes. There are two cases:

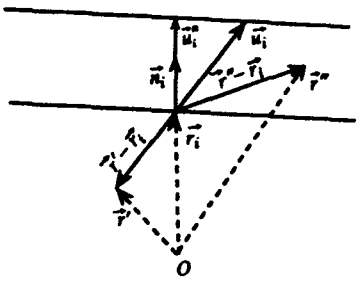


Figure 10:

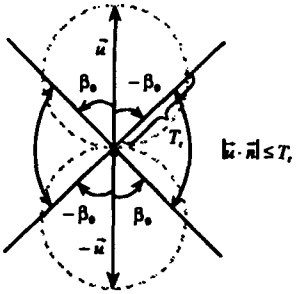


Figure 11:

Case 1: S' does not intersect the border of S, in which case the FOE is in S'.

Case 2: S' touches the border of S, in which case the FOE could be outside S.

It should be clear that if there is no rotation, then S' will always contain the FOE or give the direction of the FOE—i.e. the direction towards which we need to rotate. The size of S' depends on the distribution of features.

We can investigate the performance of the voting scheme in the presence of rotation. In particular we can ask how large area S is when rotation is present. It has been shown that this depends on the angle θ_ω between the direction of translation and the axis of rotation as well as on the rotation-to-translation ratio k. In particular, θ_ω distorts area S' and k enlarges it as it grows. The interested reader can consult [16].

5.4.2 Correctness of voting

The normal flow (as well as the actual flow) is very small in the region close to the FOE, and in the directions close to orthogonal to the directions of the flow. Consequently, even when only translation is present, in order to avoid inaccuracies that might arise in the estimated direction of the normal flow—numerical manipulation of very small quantities is unstable—we are going to discard any normal flow whose magnitude is less than some threshold T_t. Later, it will turn out that choosing this threshold greatly facilitates the geometrical analysis of the technique. Considering an actual flow $\vec{u}$ at a point A (see Figure 11) we can compute the locus of gradient directions $\vec{n}$ along which the normal flow (i.e. the projection of $\vec{u}$ on $\vec{n}$) is bigger than the threshold T_t. In Figure 11 they are all directions inside angle BAC defined by $\beta_0 = \arccos\frac{T_t}{\|\vec{u}\|}$ for $\frac{T_t}{\|\vec{u}\|} \leq 1$, or there are no such directions for $\frac{T_t}{\|\vec{u}\|} > 1$.

We now develop a condition that needs to be satisfied in order for voting at a point to be correct in the presence of rotation.

Voting will clearly be correct only if the direction of the translational normal flow is the same as the direction of the actual normal flow, that is when

$$(\vec{n}\cdot\vec{u}_t)(\vec{n}\cdot\vec{u}) > 0 \qquad (2)$$

In addition, since we consider only normal flows greater than threshold, we need

$$|\vec{n}\cdot\vec{u}| > T_t \qquad (3)$$

Inequality (2) becomes

$$\begin{aligned}(\vec{n}\cdot\vec{u}_t)(\vec{n}\cdot\vec{u}) &= (\vec{n}\cdot\vec{u}_t)(\vec{n}\cdot\vec{u}_t+\vec{n}\cdot\vec{u}_R) = \\ &= (\vec{n}\cdot\vec{u}_t)^2+(\vec{n}\cdot\vec{u}_t)(\vec{n}\cdot\vec{u}_R) > 0\end{aligned} \qquad (4)$$

So, if we set $|\vec{n}\cdot\vec{u}_R| = T_t$, then there are two possibilities: either $|\vec{n}\cdot\vec{u}|$ is below the threshold, in which case it is of no interest to voting, or the sign of $\vec{n}\cdot\vec{u}$ is the same as the sign of $\vec{n}\cdot\vec{u}_t$. In other words, if we can set the threshold equal to the maximum value of the normal rotational flow, *then our voting will always be correct.* But at point $\vec{r}$ of the sphere the rotational flow is

$$\begin{aligned}|\vec{n}\cdot\vec{u}_R| &\leq \|\vec{n}\|\cdot\|\vec{u}_R\| = \|\vec{u}_R\| = \|\vec{\omega}\times\vec{r}\| = \\ &= \|\vec{\omega}\|\cdot\|\vec{r}\|\cdot|\sin(\angle\vec{\omega},\vec{r})| \leq \|\vec{\omega}\|\end{aligned}$$

Thus if we choose $T_t = \|\vec{\omega}\|$, then the sign of $\vec{n}\cdot\vec{u}$ (actual normal flow) is equal to the sign of $\vec{u}_t\cdot\vec{n}$ (translational normal flow) for any normal flow of magnitude greater than T_t.

5.4.3 The case of dominant rotation

Although the technique described in this paper was derived to solve the problem of kinetic stabilization it turns out that it has general applicability. It can be modified to handle the case of dominant rotation with translation.

Consider a pattern of optic flow in the case of pure rotation. On a spherical retina the optic flow will correspond to vectors tangent to the circles around the axis of rotation $\vec{\omega}$. The point at which the axis of rotation passes through the image will be called the AOR. If there is circular optic flow in the image (due to pure rotation) the center of all the circles is the AOR. If we take an arbitrary optic flow vector $\vec{u}_R$ at the point $\vec{r}_i$ then we can say that a point $\vec{r}$ is a candidate for the AOR if

$$(\vec{r}_i\times\vec{u}_R)\vec{r} < 0.$$

This inequality expresses the fact that the feature point and the flow vector at the point span the plane p which cuts the sphere in two hemispheres where one contains all possible candidate points for the AOR (and all of them satisfy the previous inequality). Furthermore, all possible positions of the AOR lie on the great circle which is normal (on the sphere) to the great circle which is the intersection of the plane p and the image sphere. In

other words if we replace $\vec{u}_R$ with the normal flow $\vec{u}_R^n$ the inequality will still hold.

Very similar reasoning applies in the case of a flat retina (perspective projection). Given an optic flow (u, v) at the feature point (x_i, y_i) all possible candidate points for the AOR are on the right of the line passing through (x_i, y_i) and parallel to (u, v). Furthermore, they all lie on the line normal to (u, v) and originating at (x_i, y_i). In other words candidate points (x, y) for the AOR satisfy the inequality

$$((u, v, 0) \times (x - x_i, y - y_i, 0))(0, 0, 1) < 0.$$

This inequality indicates that the z component of the vector product of the optic flow vector and the difference of the candidate AOR point and the feature point must be negative. As in the case of a spherical retina this holds even when the optic flow (u, v) is replaced by the normal flow (u^n, v^n). As was done in the case of translation, voting can be performed. Points with maximum votes are candidates for the AOR. If a minimum is sought then the opposite direction will be found. If the area is closed then the AOR is localized as before; otherwise its general direction will be indicated by the area with maximum votes.

An analysis (on a spherical retina) similar to the one performed for the case of dominant translation can be performed again. This time, however, the threshold should be set to $T_t = \tau = \frac{R}{\|\vec{t}\|}$ (time to collision). If the magnitude of the normal flow is greater then T_t then it *must have the same sign* (and direction) as rotational normal flow.

When $\vec{\omega}$ and $\vec{t}$ are parallel the angular radius of the uncertainty region is equal to θ_{r_0} where $\cot\theta_{r_0} = \frac{\|\vec{\omega}\|}{\|\vec{t}\|}R$. The difference in the angular radii of the uncertainty areas around the FOE and the AOR is that the tangent is replaced by the cotangent. When $\theta_\omega > 0$ the uncertainty area around the AOR changes shape in a similar manner as the uncertainty area around the FOE. It extends in the direction $\vec{\omega} \times \vec{t}$ with the growth of θ_ω and gets closer to the AOR in the opposite direction.

6 Active Detection of Independent Motion

Among the more significant papers devoted exclusively to detecting moving objects is [32] by Thompson and Pong. It recognizes the difficulty of motion detection using only visual information in the form of optic flow, and considers additional constraints that may have to be applied for motion detection, e.g. knowledge about camera motion, moving object tracking, and information about scene depth. Though it presents a good discussion of the various trade-offs involved, all techniques proposed still depend on the computation of the optic flow.

The two approaches that are closest to the technique described here (emphasizing qualitative techniques for particular situations) are [17] and [27]. In [17] Bhanu et al. identify a fuzzy FOE (see also [14]) and propose a rule-based qualitative analysis of the motion of scene points. However, this requires point correspondences

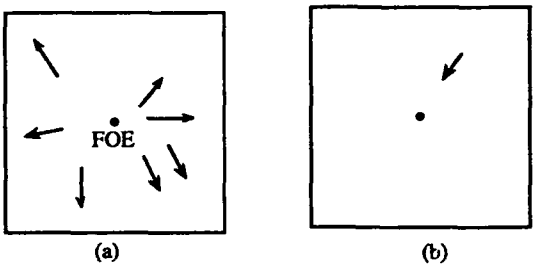


Figure 12: (a) If the observer translates along its optical axis, then the normal flow field has the property that it points away from the origin (FOE) at every point. This normal flow field is as expected and it does not signify independent motion (although it might exist). (b) There exist values of the normal flow that do not point away from the FOE. They are not as expected and thus signify independent motion.

that are difficult to obtain in general and involves considerable "high-level" (and hence expensive) reasoning, which would seem to be inappropriate for the relatively "low-level" task of motion detection. In [27] Nelson gives motion detection techniques based on normal flow and pattern recognition that can be used in situations when the observer motion is specific, and when the object motion changes rapidly in comparison with the changes in camera motion (termed "animate vision"; see also [10]).

The basis of the technique described here lies in deviations from expectations. If the observer moves in a stationary environment then he/she expects to receive a normal flow field that obeys some properties (see Figure 12). If there exist independently moving visible objects in the scene then some of these properties will not hold in parts of the normal flow image; these unexpected "anomalies" signify the existence of independent motion.[5] However, it is possible that the normal flow field appears as expected while there still exists independent motion. In the sequel we will examine the problem in more detail.

The motion field and hence the optic flow is due to the motion of the observer (inducing a flow $\vec{u}^{eg}$) and the motion of independent objects in view, inducing a flow $\vec{u}^{ind}$. Then the normal flow at every point is: $v_n = u_n^{eg} + u_n^{ind}$, where u_n^{eg}, u_n^{ind} are the normal components of $\vec{u}^{eg}$ and $\vec{u}^{ind}$ respectively.

We consider the case where the motion of the observer is translation (if there is rotation, the observer's inertial sensors can provide it; then we can derotate the normal flow field and thus consider only translation). Also, the previous algorithm (Section 5) provides the FOE (or an area containing it). To simplify the exposition we first assume that the FOE is a known point but we can easily generalize to the case where the FOE lies in an area S.

To make the image acquisition active, we assume that the camera can be given very small translation, whose

[5] This principle of deviations from expectations and anomalies is very powerful and can be used in many other situations.

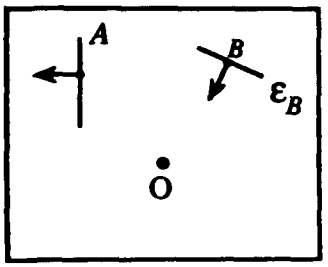


Figure 13:

net result is a momentary shift of the FOE in the image. We also assume that this can be done in a controlled manner, so that the FOE can be moved to a desired position(with a given accuracy). These small shifts are called *jitters*.

The engineering basis for using the "jitter" is that the shift in the FOE helps in motion detection (on the basis of purely geometric considerations, to be discussed), while the fact that it causes only momentary and controlled displacement about an equilibrium position eliminates the need for point correspondence. We assume that the motors responsible for moving the camera have the dynamic control capability needed for producing the jitter. We believe that with a suitably designed camera system this should be possible [6]. We do not concern ourselves here with the details of the motor control involved but consider instead only the effect of the resulting shifts of the FOE. Note that the "jitter" does not effect the dominant motion of the observer (i.e. that of the mobile platform). Thus its effect on the image flow is only "additive". That is, the image flow pattern due to the egomotion is modified by the addition of the flow due to the jitter at a point. This constrains the nature of the changes to the flow that can be brought about, as will be explained later.

6.1 The computational theory

Consider Figure 13, which represents the normal flow at two points A, B with O being the FOE. Clearly, if the normal flow points towards the FOE (i.e. the FOE lies in the half plane defined by the line normal to the flow), then this particular point (B) is moving independently of the sensor. If, however, the normal flow points away from the FOE (as in A), this could be due to egomotion or to a combination of egomotion and independent motion. Thus further constraints need to be applied to always be able to detect independent motion. At this juncture additional information from the image sequence could be used, for example, the value of u_n, but in accordance with our goal of devising a strategy that uses only the "sign" of u_n, we have to define some additional activity that may make motion detection possible. It is easy to see that the following conditions are necessary and sufficient for detecting independent motion at a point for a particular position of the FOE.

(a) $\vec{u}_n^{\text{ind}}$ points toward the FOE.

(b) The length of $\vec{u}_n^{\text{eg}}$ is less than the length of $\vec{u}_n^{\text{ind}}$.

In general, the conditions above will not be satisfied at every point in the image. The only "tool" that we allow ourselves at this point is shifting of the FOE by small "jitters" as explained earlier. The question we ask then is, what *exploratory action* (a sequence of shifts of the FOE) will guarantee motion detection at all points of the image. One condition that any exploratory activity guaranteeing "completeness" will have to satisfy is established by the following observation: If $O_1, O_2, \ldots, O_k$ is a sequence of new FOE locations (formed by an exploratory action), and the convex hull of the set of points $\{O_1 \ldots O_k\}$ encloses the entire region of interest, then complete detection of independent motion is guaranteed. This constitutes a necessary condition for the completeness of detection.

We can also observe that if the region of interest is the entire rectangular image, and the FOE is shifted at least to the four corners of the image, then the necessary condition for guaranteeing detection is satisfied. Up to now we were mostly concerned with condition (a). Before we establish conditions under which both conditions (a) and (b) are satisfied, let us consider condition (b). If it is violated, then the length of $\vec{u}_n^{\text{eg}}$ is larger than the length of $\vec{u}_n^{\text{ind}}$, or $\|\vec{u}_n^{\text{eg}}\| > \|\vec{u}_n^{\text{ind}}\|$. Any exploratory action (since we have no control over $\vec{u}_n^{\text{ind}}$) would attempt to decrease $\vec{u}_n^{\text{eg}}$. The following two exploratory activities attempt to satisfy condition (b).

If the point of interest is point $A(x_A, y_A)$ then we can either move the FOE close to A, or decrease the angle between the line connecting the FOE to point A and the gradient of the image at point A. The first action decreases the flow due to egomotion (egomotion flow at the FOE is zero) and the second action decreases the normal flow due to egomotion.

6.2 The algorithm

For a typical robot task, detecting the motions of objects that are small, distant, or slow is not very important. On the other hand, detecting the motions of objects that are large, close, or fast may be critical for the robot, and any useful motion detection strategy should guarantee the detection of such motions.[7]

[6] After all, human eyes are perpetually active and can be moved very efficiently with the help of extensive groups of muscles. Any artificial system that purports to emulate human performance—at least in achieving navigational goals (for example)—should have similar "active" capabilities [10].

[7] For example, for safe navigation a mobile robot needs to detect any sharp changes in nearby objects that are large enough to be important (e.g. another robot or a human that may move across its path), while other moving objects may not be of immediate interest if they are distant (e.g. they will not affect the robot's planned path) or if they are too small (e.g. a fly). Of course this may represent only a typical scenario; under other circumstances or for other missions all motion may be critically important, and it would be justified to pay the cost, which consists of increased exploratory activity (careful scanning of the scene) or a decrease in the overall speed of the robot. An analogy from the biological world can easily be made. When a deer or other animal senses danger it slows down or even stops completely and looks around care-

There is obviously a trade-off involved between the activity required and the parameters that describe the sensitivity of motion detection under different conditions. Computation of normal flow proceeds in real time. Normal flows pointing toward the FOE are classified as moving independently. Additional activity by the observer (moving the FOE at least in the four image boundaries) may uncover additional independently moving points.

If we consider a point $A(x, y)$ on the image plane, then the length of the flow $\vec{u}^{eg}$ is $\|\vec{u}^{eg}\| = \frac{W}{Z} \cdot r$, where r is the distance of A from the FOE. Assume further that the detection paradigm is such that for the point of interest A at least one position of the FOE lies within distance d from A. That is $r < d$ for point A. The length of the corresponding egomotion flow is thus $\|\vec{u}^{eg}\| < \frac{W}{Z} \cdot d$ and consequently the normal egomotion component obeys

$$\|u_n^{eg}\| < \frac{W}{Z} \cdot d$$

For independent motion to be detected, we need both conditions (a) and (b) to be satisfied. One way to guarantee that (a) will be satisfied is to move the FOE to a new position so that A will be inside the segment defined by the two FOE positions. For condition (b) to be satisfied we need (worst case) that

$$\begin{array}{ll} & \|\vec{u}_n^{ind}\| > \|\vec{u}_n^{eg}\| \\ \text{so that} & \|\vec{u}_n^{ind}\| > \frac{W}{Z} d \\ \text{or} & d < \|\vec{u}_n^{ind}\| \frac{Z}{W} \end{array}$$

If the above inequality is satisfied, then we are guaranteed to detect independent motion at point A.

In the above inequality, d basically represents the cost involved in detecting motion using an exploratory strategy that guarantees detection. Obviously the cost of the exploration decreases (i.e. d increases) when the time to collision with the environment is small (large depth, small W). On the other hand, if $\|\vec{u}\|$ is the smallest retinal motion (due to independent 3-D motion) that can be detected, then

$$d = \|\vec{u}\| \cdot \frac{Z}{W}.$$

If $\|\vec{u}^{ind}\| < \|\vec{u}\|$, then there is no guarantee of detection.

This formalizes the earlier intuitive discussion and also indicates a way to control the performance of the motion detection strategy. At any stage a higher precision (lower $\|\vec{u}\|$) can be achieved without changing the exploratory action (parameterized by d), but by decreasing the dominant speed of the robot.

When the purpose of motion detection is to serve as an early warning system to detect independently moving objects in the scene it is not necessary to guarantee the detection of all moving (feature) points. The detection of a few moving points (that satisfy some criteria, to eliminate "false alarms") should suffice since it can trigger a more detailed analysis (perhaps over a narrower region), again depending on the task at hand. We show how the cost of motion detection may be dramatically reduced when the requirement is to guarantee detection of a compact moving object of at least a minimum projected size. In particular, the cost of the exploratory activity can be linked to the minimum (image) diameter of the objects of interest.

As discussed earlier, it may be reasonable to assume that the boundary of the image of a compact object in the scene forms a closed contour. In particular, this implies that all the points on the boundary of the object are features, and would be successful candidates for our motion detection paradigm provided the projected motion $\vec{u}^{ind}$ is sufficiently large ($\|\vec{u}^{ind}\| > \|\vec{u}\|$). We define the diameter ϵ of an arbitrary object as the diameter of the largest circle that can be inscribed in the closed contour that forms the boundary of the projected image of the object. Now it becomes clear that any exploratory paradigm that "covers" the image so that it guarantees the detection of all points distance ϵ apart will guarantee the detection of at least a few points on an object having features. The points are guaranteed to be detected by an appropriate sequence of FOE shifts as discussed earlier. Moreover, because the boundaries of objects are locally smooth, the points thus detected will be clustered together, so that it may be possible to eliminate false alarms arising due to various noise sources that result in isolated points appearing to have independent motions. In practice, the presence of larger features on the objects, e.g. lines separating regions, would make the detection even easier. Thus the effort in the exploratory activity can be reduced when the objects of interest have image diameters greater than some threshold and when there need be no guarantee of detecting objects having projected diameters below that threshold. This is most appropriate when the purpose of the robot is such that larger and nearer objects are more interesting than smaller and farther objects, as may be true for many typical robot tasks such as safe navigation in a dynamic environment. However, at any stage the precision of the detection can be increased by decreasing the diameter and threshold using closer FOE shifts in the exploratory action.

7 Estimating 3-D Motion

Assume an observer imaging an object moving in an unrestricted rigid manner. The motion of the object can be described as the sum of a rotation plus a translation. We can choose a point through which the rotation axis passes; this gives a unique rotation and translation describing the rigid motion (in general there are infinitely many combinations of rotations and translations describing the same rigid motion). In many visual tasks we are only interested in the translation of the moving object and we need no information about how it rotates around itself. This section describes how we can estimate the direction of the object's translation without being able to recover the rotation using a technique based on normal flow. Assume that the object is translating with velocity $\mathbf{V} = (U, V, W)$ and rotating with angular velocity $\Omega = (A, B, C)$ around a point $P = (X_0, Y_0, Z_0)$ on the

fully, alert to the slightest movement (and may "jump" even for a falling leaf), whereas normally it is less sensitive to the motions around it. It would be desirable to equip a robot with a similar mechanism for motion detection that would have a variable level of sensitivity.

object. Point P is on the object and its exact choice will be made clear later.

Point P is visible in the image ($p = (x_0, y_0)$) and we attach a coordinate system onto the object at point P with axes parallel to the observer's coordinate system. We express the motion of the object in this "object-based" coordinate system. The velocity of a point Q on the object is

$$\mathbf{V} = \begin{bmatrix} U \\ V \\ W \end{bmatrix} + \Omega X \begin{bmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{bmatrix}$$

Then the normal flow v_n along direction (n_x, n_y) at point (x, y) is $v_n = un_x + vn_y$

where (u, v) is the motion field. Expressing (u, v) in terms of 3-D motion we get

$$v_n = \frac{W}{Z}\left(n_x\frac{U}{W} + n_y\frac{V}{W}\right) - \frac{W}{Z}(n_x x + n_y y)$$

$$-A(y - y_0)(xn_x + yn_y) +$$
$$+B(x - x_0)(xn_x + yn_y) +$$
$$+C[(x - x_0)n_y - (y - y_0)n_x] +$$
$$+\frac{Z - Z_0}{Z}(Bn_x - An_y),$$

or $$\frac{n_x}{v_n}\frac{U}{W} + \frac{n_y}{v_n}\frac{V}{W} - K\frac{Z}{W} = \frac{n_x}{v_n}x + \frac{n_y}{v_n}\cdot y$$

where

$$\begin{aligned} k = 1 \;&+\; A(y - y_0)\left(x\frac{n_x}{v_n} + y\frac{n_y}{v_n}\right) \\ &-\; B(x - x_0)\left(x\frac{n_x}{v_n} + y\frac{n_y}{v_n}\right) \\ &-\; C(x - x_0)\frac{n_y}{v_n} \\ &+\; C(y - y_0)\frac{n_x}{v_n} \\ &-\; \frac{Z - Z_0}{Z}\left(B\frac{n_x}{v_n} - A\frac{n_y}{v_n}\right) \end{aligned}$$

Consider a small patch of the image around point $p = (x_0, y_0)$ and let us assume that the average depth there is Z_{av}. If we add the quantity $k\left(\frac{Z}{W}\right) - \frac{Z_{\text{av}}}{W}$ to both sides of the above equation, we get

$$\frac{n_x}{v_n}\frac{U}{W} + \frac{n_y}{v_n}\frac{V}{W} - \frac{Z_{\text{av}}}{W} = \frac{n_x}{v_n}x + \frac{n_y}{v_n}y + \left(k\frac{Z}{W} - \frac{Z_{\text{av}}}{W}\right)$$

One can verify that the mean of the last term in the equation above is zero (assuming that the mean of x is x_0 and of y, y_0).

We can thus consider several linear equations:

$$\frac{n_x}{v_n}\frac{U}{W} + \frac{n_y}{v_n}\frac{V}{W} - \frac{Z_{\text{av}}}{W} = \frac{n_x}{v_n}x + \frac{n_y}{v_n}y$$

in the neighborhood around P. Solution of the system provides the FOE.

The reader must have realized that it was the choice of the coordinate system in which we expressed the motion that allowed us to isolate the translational part of the problem. Since P is the center of rotation, the rotational flow at point $p = (x_0, y_0)$ is zero. In other words the above equation is exact at point (x_0, y_0) and approximate in its neighborhood. The error terms, however, have zero mean. This provides the potential for robust estimation. The time to collision is also estimated.

It is, however, clear that the technique for addressing the passive navigation problem (Section 5) cannot be used for the 3-D motion of an object estimation problem (while they are both the same problem if considered as general recovery problems). For example, voting for the values of normal flow produced by the motion of an object can provide a very large solution area.

8 Obstacle Avoidance—Relative Depth

One of the most elementary forms of navigation is obstacle avoidance by a moving, compact sensor. It is a prerequisite, however, for many more complex abilities since any system performing a more complicated task must avoid obstacles in the process. Obstacle avoidance is thus one specific problem for which a general solution is highly desirable. In this context, a general solution refers to a system that works effectively in a wide range of real environments. This implies, among other things, that the system performance does not depend upon artificial constraints on the nature of objects in the environment such as assuming planar or smoothly curved surfaces, rigid or unmoving objects, mathematically uniform textures, and so forth.

The concept of "obstacleness" is a relative one. When we move about in our environment, every object might represent a potential obstacle, depending on its position and our direction of motion, and depending on its size. In addition, time plays an important role. When we move towards a building, the building itself represents a potential obstacle if our intent was to go beyond it. In other words, an object represents an obstacle if the observer is on a collision course with it, its size is comparable to the observer's size and the time to collision is smaller than some value which depends on the particular aspects of the problem under consideration.

Thus, we consider the problem of obstacle avoidance as synonymous to the problem of computing the times to collision to different parts of the scene, or finding relative depth at places of interest. This section is devoted to computing time to collision and relative depth from normal flow. The technique of the previous section will be used. We consider the most general case, where an object in view is moving rigidly (rotation plus translation).

8.1 Computing time to collision for a moving object

Recalling the last equation of the previous section, which is exact at the position $p = (x_0, y_0)$, we have

$$v_n = \frac{W}{Z}\left(n_x\frac{U}{W} + n_y\frac{V}{W}\right) - \frac{W}{Z}(x_0 n_x + y_0 n_y)$$

with all terms defined as previously. If $\frac{U}{W}, \frac{V}{W}$ is already computed, the quantity $\frac{Z}{W}$ is directly available.

8.2 Computing relative depth

Assume two objects A and B moving in a rigid manner, while an active observer has the task of finding which one is closer. The camera is active and can undergo a short abrupt motion along its optical axis. Let us assume that at time t_1 the camera is stationary and then it moves with velocity W_c at time t_2. Assuming that the velocities of the two objects remain unchanged during the time interval $[T_1, t_2]$, we obtain (as in Section 7) for times t_1 and t_2

$$\frac{Z^A_{\mathrm{av}}}{W_A} = A_1 \qquad \frac{Z^B_{\mathrm{av}}}{W_B} = B_1$$
$$\frac{A^A_{\mathrm{av}}+W_A dt}{W_A+W_C} = A_2 \qquad \frac{Z_{\mathrm{av}}+W_B dt}{W_B+W_C} = B_2$$

where A_1, B_1, A_2, B_2 are known.

From these equations, assuming that dt is very small, we obtain

$$\frac{Z^A_{\mathrm{av}}}{Z^B_{\mathrm{av}}} = \frac{A_1 A_2 (B_2 - B_1)}{B_1 B_2 (A_2 - A_1)},$$

and hence relative depth.

9 Visual Pursuit [8]

In a general three dimensional visual pursuit system, we find an agent, whose motion is under the control of our system; a camera, which is used to generate useful visual information to control the agent; and an object, which may be moving. If we could find the three dimensional positions and motion parameters [19] [25] of the camera, the object, and the agent, it would be a simple arithmetic problem to predict and guide the collision of the agent with the object. However, we shall show here that it is not necessary to recover these parameters.

When we try to solve the visual pursuit problem through 3D recovery, we estimate much more than we actually need in order to perform this generic visual task. Taking a purposive viewpoint we develop a robust, qualitative solution to the problem that does not require correspondence or full 3D recovery.

There are two general cases of the pursuit problem. The camera can be mounted separately from the agent and the object, or the camera can be mounted on the agent or the object. In a situation where a human agent pursues a flying ball, both of these problems are involved. The "camera" is mounted on the agent (the human's body) which is intended to collide with the object. When the human is sufficiently close to the ball, the "camera", which is mounted on the head, is independent of the agent (the hand, possibly carrying a tool such as a bat), and the hand is to collide with the ball. Thus the solution of both problems would provide a theoretical basis for an integrated mobile "hunting" system, or for a baseball player!

From a mathematical viewpoint it is equivalent whether the camera is mounted on the agent or the camera is mounted on the object. In such systems the collision is solely determined by the relative motion of the object and the agent, and it is equivalent whether we are controlling the motion of the entity that the camera is mounted on, or the entity that is moving separately from the camera. However, they may have different applications. An example of the case when the camera is mounted on the agent is an airplane that is attacking a target. An example of the case when the camera is mounted on the object is a camera that is guiding a plane to land near the camera.

Let us assume a Cartesian coordinate system with its origin at the focus of the camera, with the z-axis pointing towards the general direction of the agent and the object, such that both the object and the agent are in the full view of the camera.

Assume that the agent is located at $(X_s, Y_s, Z_s)^T$ with a velocity of $(V_{xs}, V_{ys}, V_{zs})^T$, and the object is located at $(X_o, Y_o, Z_o)^T$ with a velocity of $(V_{xo}, V_{yo}, V_{zo})^T$. If the agent or the object is also rotating at the time, we can choose the rotation axis to go through visible points on the surface of the agent or the object, chosen such that the rotation parameters are irrelevant in the prediction and guidance of a collision. However, for simplicity we assume that the motion is instantaneously translational. In the general case the analysis remains essentially the same, but the formulae become more complicated.

The agent and the object will collide after time t provided that

$$t = \frac{X_s - X_o}{V_{xo} - V_{xs}} = \frac{Y_s - Y_o}{V_{yo} - V_{ys}} = \frac{Z_s - Z_o}{V_{zo} - V_{zs}} > 0 \qquad (5)$$

If the projection of the agent (i.e. a point of it) on the image plane is (x_s, y_s), and the projection of the object is (x_o, y_o), assuming unit focal length and perspective projection, we have

$$x_s = \frac{X_s}{Z_s} \qquad (6)$$
$$y_s = \frac{Y_s}{Z_s} \qquad (7)$$
$$x_o = \frac{X_o}{Z_o} \qquad (8)$$
$$y_o = \frac{Y_o}{Z_o} \qquad (9)$$
$$v_{xs} = \frac{V_{xs}}{Z_s} - x_s \frac{V_{zs}}{Z_s} \qquad (10)$$
$$v_{ys} = \frac{V_{ys}}{Z_s} - y_s \frac{V_{zs}}{Z_s} \qquad (11)$$
$$v_{xo} = \frac{V_{xo}}{Z_o} - x_o \frac{V_{zo}}{Z_o} \qquad (12)$$
$$v_{yo} = \frac{V_{yo}}{Z_o} - y_o \frac{V_{zo}}{Z_o} \qquad (13)$$

where (v_{xs}, v_{ys}), (v_{xo}, v_{yo}) is the flow produced by the agent and the object at points (x_s, y_s) and (x_o, y_o), respectively. Combining (6–9) with (5), we obtain the following relation for the prediction of collision:

$$t = \frac{x_s Z_s - x_o Z_o}{V_{xo} - V_{xs}} \qquad (14)$$
$$= \frac{y_s Z_s - y_o Z_o}{V_{yo} - V_{ys}} \qquad (15)$$
$$= \frac{Z_s - Z_o}{V_{zo} - V_{zs}} \qquad (16)$$

[8] This section demonstrates that depth recovery is not necessary for motion coordination problems.

$$> 0 \quad (17)$$

We call (14–17) the *Visual Constraints of Collision*. The visual pursuit problem is solved if we can guide the system to satisfy these constraints. Using the processes of Sections 7 and 8 we can estimate the locomotive intrinsics (i.e. the direction of translation and the time of collision). In what follows, we solve the visual pursuit problem in the case when the camera is mounted on the object, using only the signs of the three locomotive intrinsics, and then we present a solution in the case when the camera is mounted separately to supervise the agent, using the locomotive intrinsics, relative depth, and the direction of motion.

9.1 Camera mounted on the object

The problem is equivalent whether the camera is mounted on the agent or on the object. For simplicity here we assume that the camera is mounted on the object and that we can control the velocity of the agent. We choose a Cartesian coordinate system with its origin at the focus of the camera, and with its z-axis pointing towards the general direction of the agent, such that the agent is in the full view of the camera.

As the camera is mounted on the object, the coordinates of the object on the image plane are zero, as is its velocity. We have $(X_o, Y_o, Z_o)^T = 0$ and $(V_{xo}, V_{yo}, V_{zo})^T = 0$, as well as $(x_o, y_o) = 0$ and $(v_{xo}, v_{yo}) = 0$. In the following, when we write Z_o and Z_s, we always mean $E(Z_o)$ and $E(Z_s)$ (i.e. the average depth around the neighborhood), unless otherwise specified. Thus (14–17) can be simplified to

$$t = -\frac{x_s Z_s}{V_{xs}} \quad (18)$$

$$= -\frac{y_s Z_s}{V_{ys}} \quad (19)$$

$$= -\frac{Z_s}{V_{zs}} \quad (20)$$

$$> 0 \quad (21)$$

From (18) and (20) we have $x_s = V_{xs}/V_{zs}$. From (19) and (20) we have $y_s = V_{ys}/V_{zs}$. Thus if we draw a line from the origin through the focus of expansion $(V_{xs}/V_{zs}, V_{ys}/V_{zs})$, or the first two locomotive intrinsics, on the image plane, we have a set of all the points that will collide with the origin. In order to collide the agent with the object, we should control the motion of the agent so that the focus of expansion lies inside the image of the agent. The third locomotive intrinsic Z_s/V_{zs} is the negative of the time to collision (see (21)). Note that since $t > 0$, the third locomotive intrinsic should be negative for the collision to occur. In this case, $V_{zs} < 0$, that is the agent should be coming closer to the camera.

Since we have an active camera, for simplicity we can rotate the camera such that the image of the agent will be in the center. The agent will collide with the object if we can keep the focus of expansion at the origin and keep an expanding pattern of normal flows.

If the focus of expansion is not at the origin, we can devise a control strategy to guide the focus of expansion towards the origin of the image plane according to the signs of the three locomotive intrinsics, indicating whether the velocity of the agent needs to be increased or decreased at any time instant.

- If $Z/V_z < 0$ and $V_{xs}/V_{zs} = V_{ys}/V_{zs} = 0$, a collision will occur.
- If $Z/V_z = 0$, a collision has occurred;
- If $Z/V_{zs} > 0$, the agent is going away. Decrease V_{zs} and
  - If $V_{xs}/V_{zs} = 0$, do not change V_{xs};
  - If $V_{xs}/V_{zs} > 0$, decrease V_{xs};
  - If $V_{xs}/V_{zs} < 0$, increase V_{xs};
  - If $V_{ys}/V_{zs} = 0$, do not change V_{ys};
  - If $V_{ys}/V_{zs} > 0$, decrease V_{ys};
  - If $V_{ys}/V_{zs} < 0$, increase V_{ys};
- If $Z/V_{zs} < 0$, the agent is coming closer. Do not change V_{zs} and
  - If $V_{xs}/V_{zs} = 0$, do not change V_{xs};
  - If $V_{xs}/V_{zs} > 0$, increase V_{xs};
  - If $V_{xs}/V_{zs} < 0$, decrease V_{xs};
  - If $V_{ys}/V_{zs} = 0$, do not change V_{ys};
  - If $V_{ys}/V_{zs} > 0$, increase V_{ys};
  - If $V_{ys}/V_{zs} < 0$, decrease V_{ys}.

This constitutes a qualitative paradigm for colliding the agent with the object when the camera is mounted on the object. We only use the sign of the three locomotive intrinsics. We can predict the collision, and if a collision will not occur we qualitatively control the velocity of the agent towards a state such that the agent will collide with the object.

9.2 Camera mounted separately

When the camera is mounted separately, the camera may be stationary or in motion relative to the world coordinate system. But for simplicity, we choose a coordinate system with its origin at the focus of the camera and its z-axis pointing towards the general direction of the agent and the object such that both the agent and the object are in full view of the camera. In this coordinate system, the camera is stationary, and velocity is measured relative to the camera.

9.2.1 Object coming towards the camera

The special case when the object is coming towards the camera may need to be handled differently. If we can correctly identify cases when the object is coming towards the camera, we may want to move the camera away from the pathway of the object and then proceed as usual, or when the object is small and is not destructive, we may just put the agent in front of the camera to receive the object.

We can detect whether the object is coming towards the camera using the analysis in the previous section. When the object is coming towards the camera, the focus of expansion of the object lies inside the image of the object and the third locomotive intrinsic is negative.

If we send the agent to the front of the camera, we have the case studied in the last section. It can be determined

from the time to collision of the agent and the object whether the agent is moving fast enough to intercept the object.

In the following general analysis, we assume that the object is not coming towards the camera, but moving in any other direction.

9.2.2 General case of camera mounted separately

From (14–15), we obtain

$$((x_s V_{yo} - y_s V_{xo}) - (x_s V_{ys} - y_s V_{xs}))Z_s - ((x_o V_{yo} - y_o V_{xo}) - (x_o V_{ys} - y_o V_{xs}))Z_o = 0 \quad (22)$$

Note that if (x_s, y_s), $(V_{xs}/V_{zs}, V_{ys}/V_{zs})$, (x_o, y_o), and $(V_{xo}/V_{zo}, V_{yo}/V_{zo})$ are a group of parallel vectors, (22) will be satisfied. Thus in the general case of a separately mounted camera, we first obtain the direction of motion of the object; then we rotate the z-axis of the camera such that it will be in the direction of the object. Then we can move the agent in the direction parallel to the direction of motion of the object. This is a group of sufficient conditions for the collision of the agent and the object when we have good control of the original position of the agent.

To satisfy (16–17), we need to find the time to collision t and make it equal to (16). Combining (10–13) with (14–17), we obtain

$$t = \frac{x_s - x_o \frac{Z_o}{Z_s}}{(v_{xo} + x_o \frac{V_{zo}}{Z_o})\frac{Z_o}{Z_s} - (v_{xs} + x_s \frac{V_{zs}}{Z_s})} \quad (23)$$

$$= \frac{y_s - y_o \frac{Z_o}{Z_s}}{(v_{yo} + y_o \frac{V_{zo}}{Z_o})\frac{Z_o}{Z_s} - (v_{ys} + y_s \frac{V_{zs}}{Z_s})} \quad (24)$$

$$= \frac{1 - \frac{Z_o}{Z_s}}{\frac{V_{zo}}{Z_o} \cdot \frac{Z_o}{Z_s} - \frac{V_{zs}}{Z_s}} \quad (25)$$

$$> 0 \quad (26)$$

If we can find a point on the agent and a point on the object which have the same normal direction (n_x, n_y), from (23–24) we find the time to collision as follows:

$$t = \frac{n_x x_s + n_y y_s - (n_x x_o + n_y y_o)\frac{Z_o}{Z_s}}{(v_{no} + (n_x x_o + n_y y_o)\frac{V_{zo}}{Z_o})\frac{Z_o}{Z_s} - (v_{ns} + (n_x x_s + n_y y_s)\frac{V_{zs}}{Z_s})} \quad (27)$$

Similar equations can be obtained if we can find two normal directions from the agent and the object which are perpendicular. Combining with (25), we have

$$\frac{V_{zs}}{Z_s} = \frac{V_{zo}}{Z_o} - \frac{(\frac{Z_s}{Z_o} - 1)(v_{no}\frac{Z_o}{Z_s} - v_{ns})}{n_x(x_s - x_o) + n_y(y_s - y_o)} \quad (28)$$

Thus, control of the agent is achieved by varying V_{xs}/V_{ys} in order to satisfy (22) and V_{zs} in order to satisfy (26) and (28). According to these equations, we can devise a system for qualitative control of the motion of the agent, so that the agent will collide with the object. This scheme can be accomplished through six sequential phases as follows. The first three phases are devised to satisfy (22). The next two phases are devised to satisfy (26) and (28). The last phase tests to see if the agent will collide with the object without further control of the agent.

1. Rotating the camera. Through the point (x_o, y_s) draw a line in the image plane with direction $(V_{xo}/V_{zo}, V_{yo}/V_{zo})$.
   - If the line goes through the origin, proceed to the next phase;
   - If the origin is on the lower left portion of the image plane, rotate the camera up and to the right;
   - If the origin is on the upper right portion of the image plane, rotate the camera downward and to the left;
2. Position the agent.
   - If the line drawn above goes through the agent, proceed to the next phase;
   - If the agent is on the lower left portion of the image plane, move the agent up and to the right;
   - If the agent is on the upper right portion of the image plane, move the agent downward and to the left;
3. Move the agent parallel to the image plane. Change the velocity of the agent, such that $V_{xs}/V_{ys} = x_s/y_s$.
   - If the agent is on the left of the object, V_{xs} and V_{ys} should be increased;
   - If the agent is on the right of the object, V_{xs} and V_{ys} should be decreased;
   - If the agent and the object collide on the image plane, do not change V_{xs} and V_{ys};
4. Positive time to collision. Proceed to the next phase after:
   - If $Z_o/Z_s = 1$, do not change V_{zs} at present;
   - If $Z_o/Z_s > 1$, adjust V_{zs} such that
   $$\frac{V_{zs}}{Z_s} > \frac{Z_o}{Z_s} \cdot \frac{V_{zo}}{Z_o};$$
   - If $Z_o/Z_s < 1$, adjust V_{zs} such that
   $$\frac{V_{zs}}{Z_s} < \frac{Z_o}{Z_s} \cdot \frac{V_{zo}}{Z_o};$$
5. Move the agent perpendicular to the image plane.
   - If (28) is satisfied, proceed to the next phase.
   - If V_{zs}/Z_s is larger in (28), decrease V_{zs};
   - If V_{zs}/Z_s is smaller in (28), increase V_{zs};
6. Predicting collision. The agent will collide with the object if the following conditions are met, or otherwise repeat from phase 1:

$$\frac{x_o}{y_o} = \frac{\frac{V_{xo}}{V_{zo}}}{\frac{V_{yo}}{V_{zo}}} = \frac{x_s}{y_s} = \frac{\frac{V_{xs}}{V_{zs}}}{\frac{V_{ys}}{V_{zs}}}$$

$$(\frac{Z_o}{Z_s} - 1)(\frac{V_{zs}}{Z_s} - \frac{Z_o}{Z_s} \cdot \frac{V_{zo}}{Z_o}) > 0$$

and

$$\frac{V_{zs}}{Z_s} = \frac{V_{zo}}{Z_o} - \frac{(\frac{Z_s}{Z_o} - 1)(v_{no}\frac{Z_o}{Z_s} - v_{ns})}{n_x(x_s - x_o) + n_y(y_s - y_o)}$$

In summary, in the case when the camera is mounted on the the object or the agent, we have devised a qualitative strategy to predict and guide the collision of the agent and the object. We only used the signs of the three locomotive intrinsics (FOE and time to contact) to qualitatively control the velocity of the agent such that the visual constraints of collision will be satisfied.

In the case when the camera is mounted separately to supervise the agent to collide with the object, we have devised a set of sufficient conditions to satisfy the visual constraints of collision. This set of sufficient conditions can be reached by a qualitative scheme of control without any exact 3D depth or velocity information of the agent and the object.

Our method can also be used to control the collision even when the object is rotating in addition to having instantaneous translational motion.

10 Recapitulation and Experiments

We have presented solutions to several problems related to visual motion using normal flow as the input. Although we have not solved the general structure from motion (sfm) problem using normal flow, we have presented solutions to various important problems that are simple applications of the sfm module. The robustness of the proposed algorithms relies heavily on the robustness of the computation of normal flow, i.e. spatiotemporal derivatives of the image intensity function. But even without using any elaborate schemes for computing the normal flow (after all, some of the techniques presented only require its sign) we have performed several experiments. We report here a few of them:

(a) **Egomotion estimation**

We have performed several experiments with both synthetic and real image sequences in order to demonstrate the stability of our method. From experiments on real images it was found that in the case of pure translation or pure rotation the method computes the focus of expansion or the axis of rotation very robustly. In the case of general motion it was found from experiments on synthetic data that the behavior of the method is as predicted by our theoretical analysis (see [16]).

Figure 14 shows one of the images from a dense sequence collected in our laboratory using an Merlin American Robot arm that translated while acquiring images with the camera it carried (a Sony miniature TV camera). Figure 15 shows the last frame in the sequence and Figure 16 shows the first frame with the solution area (where the FOE lies), which agrees with the ground truth. Figures 17 and 18 show the first and last frames in a sequence of images collected through a rotation of the sensor and provided by the University of Massachusetts for the Workshop. Figure 19 shows the first frame of the sequence with the solution area for the AOR.

(b) **Detection of independent motion**

Figure 20 shows the experimental setting for testing the algorithm for motion detection from a translating, active camera. The CCD camera is mounted on a slide to simulate pure translation, and can be given small rotations around a revolving platform to simulate the exploratory activity. The model board simulates an out-



Figure 14:



Figure 15:



Figure 16:

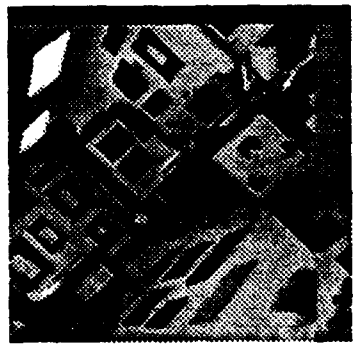


Figure 17:



Figure 18:

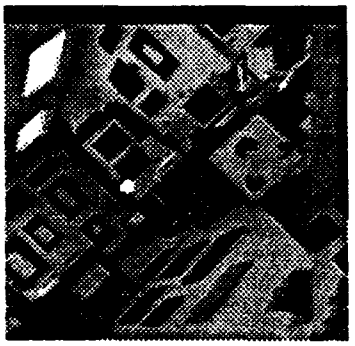


Figure 19:

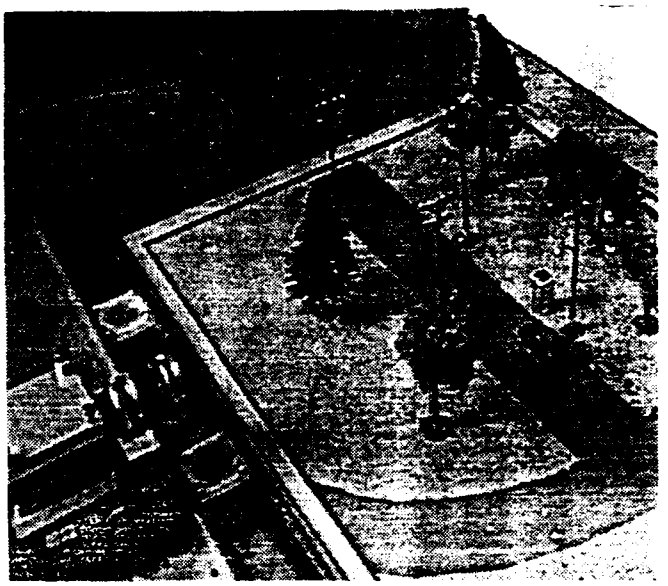


Figure 20:



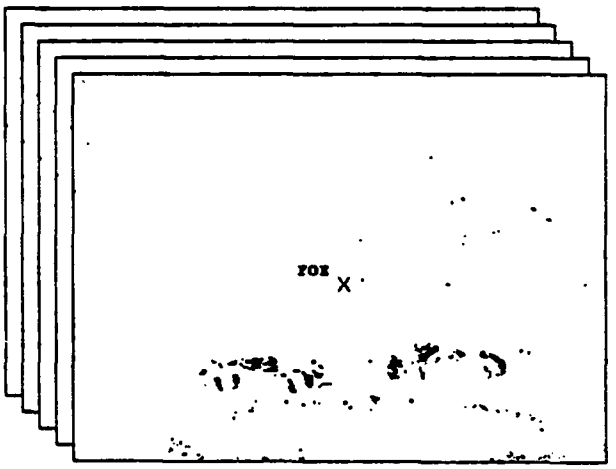


Figure 21: (a) and (b)

door scene. The image sequence is captured by Data Translation QuickCapture on a Macintosh IIci. Figures 21(a)–21(d) show the results of the experiment: (a) is a sequence of closely sampled images taken from a moving and active camera; (b) shows the output of the motion detection algorithm without any exploratory activity; (c) shows the output after four shifts of the FOE as part of a simple exploratory activity; and (d) shows the motion detection output (dark) overlaid on the image (light).

(c) Relative Depth

We have performed several experiments on both synthetic and real data in order to test the feasibility and stability of our approach. We report here some experiments on real data. The setup for our experimental work with real images consists of a CCD camera mounted on a slide so that it can purely translate along its optical axis. The camera is viewing a scene consisting of a toy ("Mrs. Potatohead") and a toy robot arm (Radio Shack). The arm (carrying the "vision" of Mrs. Potatohead) is initially placed closer to the camera. Figures 22 and 23 are taken with the camera stationary and the arm moving toward Mrs. Potatohead. Figure 24 shows the normal flow produced from the motion of the arm, using the straightforward gradient technique [19]. Figure 25 is taken after the camera has moved forward and Figure 26 shows the normal flow produced.

Using the algorithms in Sections 5 and 7, we estimated the relative depth of the toy and the arm. We computed the quantity Z/V_c, where Z is the depth of a point and V_c is the speed of the camera, and considered the median value for the arm and the toy. It was found that this



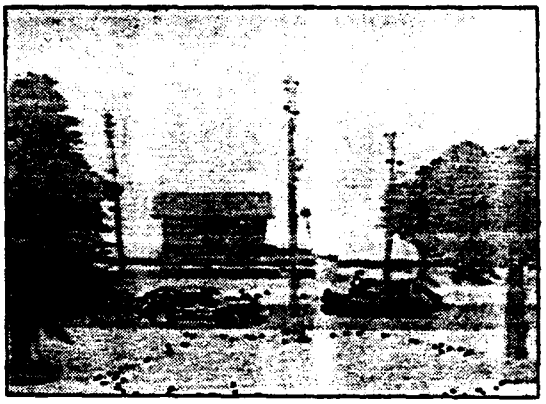


Figure 21: (c) and (d)



Figure 22:



Figure 23:



Figure 24:



Figure 25:



Figure 26:

value was 7.553544 for the arm and 9.118339 for the toy, which agrees with the ground truth.

We performed the same experiment with the arm at the same distance as the toy. We found that the value of the median relative depth (Z/V_c) was 10.230856 for the arm and 10.145772 for the toy, which again agrees with the ground truth.

11 Purposive, Behavioral, Active Vision

Vision has been studied, for the most part, as a general recovery problem, i.e. its goal has been to reconstruct an accurate representation of the visible world and its properties, for example, to recover boundaries, shape from texture, shading, motion, etc. Following this point of view, we consider the "brain"—or any intelligent system possessing vision—as consisting of vision and everything else (planning, reasoning, memory, etc.). In other words, we view the role of vision as that of creating a central database which stores accurate 3-D information about the scene. Then other cognitive processes (such as planning, for example) can access this database, extract whatever information they need and modify it to suit their needs. This central database is created by visual modules—such as the sfm module—that have been integrated in some way [3].

But if the analysis in this paper is valid, it demonstrates that we can solve many interesting problems, without creating a very accurate or full representation of the scene and its properties. Clearly, when a problem is simpler and more restricted, it is easier to solve. However, these simpler problems (in our case, simpler than the general sfm problem)—namely, passive navigation, motion detection, 3-D translation estimation, obstacle avoidance, relative depth, visual interception—are quite important and not very specific. They are generic in the sense that they have environmental invariance. In other words, developing such visual motion capabilities constitutes theoretical research. The fact that we may be able to robustly solve many less general problems—which, of course, cannot replace the reconstructive modules–demonstrates that we are capable of building machines that robustly achieve various behaviors. By putting such behaviors together, can we achieve "intelligent systems"? If this is possible, it provides an alternative way to study perception. A few publications over the past few years [6, 8, 11, 12, 15, 26, 35] have supported such an approach, which has acquired various names such as purposive, task-based, behavioral, active, animate, utilitarian, etc. In this section we attempt to describe the paradigm in more detail and we point out its drawbacks as well as its potential usefulness.

11.1 An attempt to formalize

With the realization that behavioral vision has as its goal the development of robust, non-primitive behaviors displayed by a robotic agent, we should be able to formalize the concept of behavior and the concept of an agent. At the same time we need to be able to provide a formal way of generating new behaviors and a calculus of behaviors or purposes.

If there is a similarity of this approach to old ideas of goal-based vision—where systems using knowledge at all levels, including domain-specific knowledge, were built and it turned out that many corners had to be cut and many oversimplified assumptions had to be made—it exists only in spirit. *An intelligent agent (observer) is a system that has a set of goals or purposes, at all times. To pursue these goals, it has to exhibit a set of behaviors.* Not all agents have the same purposes; some are more sophisticated than others and they display different behaviors.

It would be hard to give a general definition for an agent (or such a definition would be so general that it wouldn't be useful at an engineering level). We are surrounded by agents. They are basically entities that *interact* with the world around them and *act* appropriately in each situation. As they act and sense, they display behaviors and fulfill purposes.

Coming back to the basic question of formalizing behaviors, we realize that there is a very rich set of them. Some are primitive, others more sophisticated and others quite complex. In such situations, it is nice to be able to start from primitives, that is a set of behaviors from which all others can be constructed. But it is not at all clear which behaviors are the primitive ones.

To avoid a potential philosophical snare, we sidestep the question and we ask: how can we formalize behaviors, and then generate new and more complex ones from old ones and from learning?

A behavior is a sequence of perceptual events and actions whose task is to accomplish a goal. Visual input is received in a continuous manner and various processes (such as those described here and others) work together in order to recognize perceptual events and take appropriate action (an action could be a motion (navigation, manipulation), or a change of an internal state of the agent displaying the behavior). The problem is then to control such a system. It must be emphasized that the processes performing the visual analysis in order to recognize the perceptual events perform only partial recovery of the world, i.e. to accomplish some behaviors we do not need an accurate and full scene representation.

In abstract terms, a behavior of an agent is a system broadly known as discrete event process [7]. However, despite numerous results in the literature, there is at the present time apparently no unifying theory for the control of discrete event processes. Nor is it very clear what such a theory should accomplish. Numerous approaches to the modeling of discrete processes have appeared in the literature (Boolean models, Petri nets, formal languages, temporal logic, port automata, and flow networks).

An interesting model proposed recently [37] treats the controlled set of processes as the generator of a formal language (an automaton taking various actions) and studies how the recognizer of a specific (target) language (another machine recognizing perceptual events) may be employed as a controller, incorporating the desired closed-loop system behavior, and it is shown how to construct such a controller under some assumptions. It is also shown how, given two such controllable systems

(let's say behaviors B_1 and B_2), to create the shuffle operation $B_1 || B_2$, so that we can create more complex behaviors from existing ones. However, the main conclusion of such control theoretic work may be paraphrased by saying that "supervisors must be modeled on the task to be accomplished". In other words, there does not appear to be a general universal way to accomplish behaviors (control them) or make new ones from old ones; it appears that the problem depends on what has to be accomplished.

11.2 Object recognition

Although it is not hard to see how to study navigation in this paradigm of behavioral vision, it might seem hard to apply this point of view to recognizing objects. What would it mean to have behaviors that recognize objects?

This difficulty can be easily avoided by attempting to solve an easier problem, namely that of recognizing the function of an object[9] (there may exist many functions for a single object) that is required to accomplish the behavior under consideration (an agent always executes a behavior). So, recognition can be considered in the context of an agent performing it in an environment, while executing a behavior.

An object can fulfill a function, suit a purpose. If the agent recognizes this, it has recognized the object. In fact, it has not recognized an object in the sense that it can name it as a human would, but it has recognized it "well enough" to act on it (for example, use it, avoid it, eat it, mate with it, etc.). But in most cases, deducing an object's purpose with regard to the current behavior can be done by testing the existence of some perceptual properties of the image of the object. Usually, to find out if an object can fulfill a function we need to perform various *partial* recovery tasks. Thus, without reconstructing the world fully, we can recognize many objects to the degree that we can utilize them (examples: big and moving closer (danger), man-made, graspable, movable, of certain size, with a concavity (cup), etc.).

Although such an approach does not address all aspects of object recognition, it seems to be well suited to the design of robots.

12 Conclusions

We have presented the foundations behind a set of processes that interpret visual motion in a purposive manner. We showed that an active observer can solve a series of important problems through the use of the derivatives of the image intensity function. In particular, we presented direct solutions for the problems of kinetic stabilization (passive navigation), detection of independent motion, obstacle avoidance, relative depth and 3-D motion (translation) computation and visual interception. Although the abovementioned problems are applications of the general structure from motion problem, we addressed them as independent problems in their own right and produced solutions that depend on data which can be measured.

[9] I.e. not recognizing the object but finding out enough information about it to utilize it.

The possibility that important behaviors can be realized by the cooperation of processes that recognize perceptual events without having to create a full representation of the outside world suggests that vision can be studied as a part of a system that has purposes which translate into behaviors. This point of view opens several interesting research areas, all related to the development of intelligent visual behaviors. We have pointed out various possible formalizations for this approach, as well as the associated problems.

Research in this paradigm will become more interdisciplinary with time, since the basic premise is that vision should not be studied in isolation but as a part of an intelligent system. New questions about control arise, and the integration of vision with planning, manipulation, memory and learning will provide interesting research avenues.

Whether this behavioral vision paradigm is the natural evolution of the field is still questionable. This will certainly depend on the results that are generated. Behavioral vision addresses a normative question (what should be), i.e. how should we best design robots for a set of tasks. Reconstructive vision addresses a theoretical question (what could be), i.e. what range of possible mechanisms could exist in vision systems. The empirical question (what is), i.e. how actual biological systems are designed, is addressed by other communities(psychology, neuroanatomy, etc.), while the normative and theoretical questions are studied by computer vision. And although these three questions do not necessarily have the same answers, they are closely related.

Acknowledgements: Thanks to Ruzena Bajcsy for numerous discussions and for suggesting that an observer could be modeled as a discrete event dynamic system, and to Barbara Burnett for her help in preparing the paper and doing the artwork.

This work would not have been possible without the support of DARPA, NSF, Alliant Techsystems, Inc. and Texas Instruments, Inc.

References

[1] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Trans. PAMI*, 7:384–401, 1985.

[2] G. Adiv. Inherent ambiguities in recovering 3d motion and structure from a noisy flow field. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 70–77, 1985.

[3] J. Aloimonos and D. Shulman. *Integration of Visual Modules: An Extension of the Marr Paradigm*. Academic Press, Boston, 1989.

[4] Y. Aloimonos. Purposive and qualitative active vision. In *Proc. DARPA Image Understanding Workshop*, pages 816–828, 1990.

[5] Y. Aloimonos, I. Weiss, and A. Bandopadhay. Active vision. *Int'l. J. Comp. Vision*, 2:333–356, 1988.

[6] M.A. Arbib. Perceptual structures and distributed motor control. In V.B. Brooks, editor, *Handbook of*

Physiology: The Nervous System II. Motor Control, pages 1449–1480, 1981.

[7] R. Bajcsy. Personal communication, 1991.

[8] R. Bajcsy and P. Allen. Sensing strategies. In *Proc. U.S.-France Robotics Workshop*, 1984.

[9] D.H. Ballard. Parameter networks. *Artificial Intelligence*, 22:235–267, 1984.

[10] D.H. Ballard. Reference frames for animate vision. In *Proc. Int'l. Joint Conference on Artificial Intelligence*, pages 635–1641, 1989.

[11] R.A. Brooks. Achieving artificial intelligence through building robots. Technical Report TR 899, M.I.T., Cambridge, MA, 1986.

[12] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE J. Robotics Automation*, 2:14–23, 1986.

[13] A. Bruss and B.K.P. Horn. Passive navigation. *Computer Vision, Graphics, Image Processing*, 21:3–21, 1983.

[14] W. Burger and B. Bhanu. On computing a 'fuzzy' focus of expansion for autonomous navigation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 563–568, 1989.

[15] C.H. Chen and A.C. Kak. A robot vision system for recognizing 3-d objects in low-order polynomial time. *IEEE Trans. Systems, Man, Cybernetics, Special Issue on Computer Vision*, 1989.

[16] Z. Duriç and Y. Aloimonos. Passive navigation: An active and purposive solution. Technical Report CAR-TR-560, Center for Automation Research, University of Maryland, College Park, MD, 1991.

[17] B. Bhanu et al. Qualitative target motion detection and tracking. In *Proc. DARPA Image Understanding Workshop*, pages 370–398, 1989.

[18] C. Fermüller and Y. Aloimonos. Estimating 3-d motion from image gradients. Technical Report CAR-TR-554, Center for Automation Research, University of Maryland, College Park, MD, 1991.

[19] B.K.P. Horn. *Robot Vision*. McGraw Hill, New York, 1986.

[20] B.K.P. Horn and E.J. Weldon Jr. Computationally-efficient methods of recovering translational motion. In *Proc. International Conference on Computer Vision*, pages 1–11, 1987.

[21] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[22] L. Huang and Y. Aloimonos. Relative depth from motion using normal flow: An active and purposive solution. Technical Report CAR-TR-535, Center for Automation Research, University of Maryland, College Park, MD, 1991.

[23] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[24] H.C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proc. Royal Soc. London B*, 208:385–397, 1984.

[25] D. Marr. *Vision*. W.H. Freeman, San Francisco, 1982.

[26] H.P. Moravec. Towards automatic visual obstacle avoidance. In *Proc. IJCAI-77*, page 584, 1977.

[27] R.C. Nelson. Qualitative detection of motion by a moving observer. In *Proc. DARPA Image Understanding Workshop*, pages 329–338, 1990.

[28] R.C. Nelson and J. Aloimonos. Finding motion parameters from spherical flow fields (or the advantages of having eyes in the back of your head). *Biological Cybernetics*, 58:261–273, 1988.

[29] R. Sharma and Y. Aloimonos. Robust detection of independent motion: An active and purposive solution. Technical Report CAR-TR-534, Center for Automation Research, University of Maryland, College Park, MD, 1991.

[30] M.E. Spetsakis and J. Aloimonos. Structure from motion using line correspondences. *Int'l. J. Computer Vision*, 4:171–183, 1990.

[31] M.E. Spetsakis and Y. Aloimonos. Optimal computing of structure from motion using point correspondences in two frames. In *Proc. International Conference on Computer Vision*, pages 449–453, 1988.

[32] W.B. Thompson and T-C. Pong. Detecting moving objects. *Int'l. J. Computer Vision*, 4:39–57, 1990.

[33] R.Y. Tsai and T.S. Huang. Uniqueness and estimation of three dimensional motion parameters of rigid objects with curved surfaces. *IEEE Trans. PAMI*, 6:13–27, 1984.

[34] S. Ullman. *The Interpretation of Visual Motion*. MIT Press, Cambridge, MA, 1979.

[35] S. Ullman. Visual routines. *Cognition*, 18:97–157, 1984.

[36] A. Verri and T. Poggio. Against quantitative optic flow. In *Proc. International Conference on Computer Vision*, pages 171–180, 1987.

[37] W.M. Wonham and P.J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control Optimization*, 25:637–659, 1987.

[38] G.S. Young and R. Chellappa. 3-d motion estimation using a sequence of noisy stereo images. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 710–716, 1988.

Motion and Structure Factorization and Segmentation of Long Multiple Motion Image Sequences

Chris Debrunner[1]
Narendra Ahuja

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Abstract

This paper presents a computer algorithm which, given a dense temporal sequence of intensity images of multiple moving objects, will separate the images into regions showing distinct objects, and for those objects which are rotating, will calculate the three-dimensional structure and motion. The method integrates the segmentation of trajectories into subsets corresponding to different objects with the determination of the motion and structure of the objects. Trajectories are partitioned into groups corresponding to the different objects by fitting the trajectories from each group to a hierarchy of increasingly complex motion models. This grouping algorithm uses an efficient motion estimation algorithm based on the factorization of a measurement matrix into motion and structure components. Experiments are reported using two real image sequences of 50 frames each to test the algorithm.

1 Introduction

This paper is concerned with three-dimensional structure and motion estimation for scenes containing multiple independently moving rigid objects. Our algorithm uses the image motion to separate the multiple objects from the background and from each other, and to calculate the three-dimensional structure and motion of each such object. The two-dimensional motion in the image sequence is represented by the image plane trajectories of feature points. The *motion* of each object, which describes the rotation and translation of the object between the images of the sequence, is computed from the object's feature trajectories. If the object on which a particular group of feature points lie is rotating, the relative three-dimensional positions of the feature points, called the *structure* of the object, can also be calculated.

Our algorithm is based on the following assumptions: (1) the objects in the scene are rigid, i.e., the three-dimensional distance between any pair of feature points on a particular object is constant over time, (2) the feature points are orthographically projected onto the image plane, and (3) the objects move with constant rotation per frame. This algorithm integrates the task of segmenting the images into distinctly moving objects with the task of estimating the motion and structure for each object. These tasks are performed using a hierarchy of increasingly complex motion models, and using an efficient and accurate factorization-based motion and structure estimation algorithm.

This paper makes use of an algorithm for factorization of a measurement matrix into separate motion and structure matrices as reported in [1]. In [2], Tomasi and Kanade present a similar factorization-based method which allows arbitrary rotations, but does not have the capability to process trajectories starting and ending at arbitrary frames. Furthermore, it appears that some assumptions about the magnitude or smoothness of motion are still necessary to obtain feature trajectories. Kanade points out [3] that with our assumption of constant rotation we are absorbing the trajectory noise primarily in the structure parameters whereas their algorithm absorbs them in both the motion and structure parameters.

Most previous motion-based image sequence segmentation algorithms use optical flow to segment the images based on consistency of image plane motion. Adiv in [4] and Bergen *et al* in [5] instead seg-

Supported by DARPA and the NSF under grant IRI-89-02728, and the State of Illinois Department of Commerce and Community Affairs under grant 90-103.

[1] Currently at Martin Marietta Corporation, Denver CO 80201.

ment on the basis of a fit to an affine model. Adiv further groups the resulting regions to fit a model of a planar surface undergoing 3-D motions in perspective projection. In [6] Boult and Brown show how Tomasi and Kanade's motion factorization method can be used to split the measurement matrix into parts consisting of independently moving rigid objects.

2 Structure and Motion Estimation

In this section we describe the motion and structure parameters which are used in Section 4, and we briefly review our rotational motion and structure estimation algorithm, the details of which are presented in [1], [7], and [8]. The input to the structure and motion estimation algorithm is a set of trajectories of orthographically projected feature points lying on a single rigid object rotating around a fixed-direction axis and translating along an arbitrary path. If these constraints do not hold exactly the algorithm will produce structure and motion parameters which only approximately predict the input trajectories. Given a collection of trajectories (possibly all beginning ai d ending at different frames) for which the constraints do hold, our algorithm finds accurate estimates of the relative three-dimensional positions of the feature points at the start of the sequence and the angular and translational velocities of the object. The algorithm also produces a confidence number, in the form of an error between the predicted and the actual feature point image positions. Aside from an SVDs, the algorithm is closed form and requires no iterative optimization.

2.1 The Motion and Structure Models

We define the world coordinate system such that its x-y axes are the image plane x-y axes. The object on which the feature points are located rotates with a constant angular velocity of $\vec{\Omega}$ and an arbitrary translation described by the sequence of locations $\vec{c}_0(f)$. We define an object centered coordinate system whose z axis is parallel or anti-parallel to $\vec{\Omega}$ (such that it lies in the positive z halfplane) and which translates but does not rotate with the object. We also let $\hat{n}$ be the object coordinate system vector $[0\ 0\ 1]^T$ expressed in world coordinates. Then, letting $\vec{O}_p(f)$ be the position of point p in object centered coordinates at the time frame f is recorded, the positions $\vec{O}_p(f)$ change over time due only to the rotation of the object and can be expressed in terms of $\vec{O}_p(0)$ as $\vec{O}_p(f) = G^f \vec{O}_p(0)$ where $\omega = \hat{n} \cdot \vec{\Omega}$ and

$$\mathbf{G} = \begin{bmatrix} \cos\omega & -\sin\omega & 0 \\ \sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Since $\hat{n}$ contains only two degrees of freedom, we replace it with the two dimensional *foreshortening* vector $\hat{F}$ defined as $\mathbf{0}$ if $n_{xy} = \sqrt{n_x^2 + n_y^2} = 0$ or

$$\hat{F} = \begin{bmatrix} F_x \\ F_y \end{bmatrix} = \pm \frac{1}{n_{xy}} \begin{bmatrix} n_x\sqrt{1-n_z} \\ n_y\sqrt{1-n_z} \end{bmatrix} \tag{2}$$

otherwise, where the sign is chosen such that $F_y > 0$. Then if

$$\mathbf{H} = \begin{bmatrix} 1-F_x^2 & -F_xF_y & F_x \\ -F_xF_y & 1-F_y^2 & F_y \end{bmatrix} \tag{3}$$

and $\hat{c}_0(f) = \left[c_{0x}(f)\ c_{0y}(f)\right]^T$, the image plane coordinates of point p in frame f are (see [8])

$$\hat{P}_p(f) = \mathbf{HG}\left(\begin{bmatrix} O_{px}(0) \\ O_{py}(0) \\ O_{pz}(0)\sqrt{2-F_x^2-F_y^2} \end{bmatrix} + \hat{c}_0(f)\right) \tag{4}$$

Equation (4) is the starting point for the next section which briefly describes our algorithm for finding the motion and structure parameters.

2.2 The Algorithm

Our algorithm uses sinusoidal retrieval methods [9] which extract multi-dimensional sinusoidal signals from noisy data. In our problem, the multi-dimensional sinusoidal signal is the rotational component of the motion, and the frequency of the recovered sinusoid is ω, while the amplitudes and phases determine the structure and orientation of the points. We start by taking the differences in point positions $\hat{y}_{pq}(f) = \hat{P}_q(f) - \hat{P}_p(f)$ to eliminate $\hat{c}_0(f)$ in Equation (4). The resulting purely rotational motion can be expressed as a system of state equations

$$\begin{aligned} \hat{x}_{pq}(f+1) &= \mathbf{G}\hat{x}_{pq}(f) = \mathbf{G}^{f+1}\hat{x}_{pq}(0) \\ \hat{y}_{pq}(f) &= \mathbf{H}\hat{x}_{pq}(f) \end{aligned} \tag{5}$$

where

$$\hat{\mathbf{x}}_{pq}(f) = \begin{bmatrix} O_{qx}(f) - O_{px}(f) \\ O_{qy}(f) - O_{py}(f) \\ (O_{qz}(f) - O_{pz}(f))\sqrt{2 - F_x^2 - F_y^2} \end{bmatrix}. \quad (6)$$

The $\hat{\mathbf{y}}_{pq}(f)$ are arranged in a data matrix **D** which can be factored using a SVD into two matrices, one depending only on the rotational parameters, and the other depending only on the structure parameters. The construction of **D** and the method of finding the structure and motion from the factors of **D** is detailed in [1], [7], and [8].

The algorithm described in this section provides an efficient way to incorporate information from many frames and points into the calculation of the rotational motion and structure parameters. When there is no rotational motion, we use a simple linear least squares fit to a translational motion model (described further in [7] and [8]) to determine the motion. The following section describes how this translational motion estimation algorithm and the rotational motion estimation algorithm are used to segment the input trajectories into groups corresponding to the different rigid objects and to determine the structure and motion parameters for each object.

3 Image Sequence Segmentation and Motion and Structure Estimation

The segmentation of the feature point trajectories into groups corresponding to the differently moving 3D objects and the estimation of the structure and motion of these objects are highly interrelated processes: if the correct segmentation is not known, the motion and structure of each object cannot be accurately computed, and if the 3D motion of each object is not accurately known, the trajectories cannot be segmented on the basis of their 3D motion. To circumvent this circular dependency, we integrate the segmentation and the motion and structure estimation steps into a single step, and we incrementally improve the segmentation and the motion and structure estimates as each new frame is received.

The general segmentation paradigm is split and merge. Each group of trajectories (or *region*) in the segmentation has associated with it one of three *region motion models*, two of which describe rigid motion (the *translational* and *rotational* motion models), and the third (*unmodeled* motion) which accounts for all motions which do not fit the two rigid motion models and do not contain any local motion discontinuities. When none of these motion models accurately account for the motion in the region, the region is split using a region growing technique. When splitting a region, a measure of motion consistency is computed in a small neighborhood around each trajectory in the region. If the motion is consistent for a particular trajectory, we assume that the trajectories in the neighborhood all arise from points on a single object. Thus the initial subregions for the split consist of groups of trajectories with locally consistent motion, and these are grown out to include the remaining trajectories.

Initially all the trajectories are in a single region. Processing then continues in a uniform fashion: the new point positions in each new frame are added to the trajectories of the existing regions, and then the regions are processed to make them compatible with the new data. The processing of the regions is broken into four steps: (1) if the new data does not fit the old region motion model, find a model which does fit the data or split the region, (2) add any newly visible points or ungrouped points to a compatible region, (3) merge adjacent regions with compatible motions, (4) remove outliers from the regions.

Compatibility among feature points is checked using the structure and rotational motion estimation algorithm described in Section 2 or the translational motion algorithm described in [7]. A region's feature points are considered incompatible if the fit error returned by the appropriate motion estimation algorithm is above a threshold. We assume that the trajectory detection algorithm can produce trajectories accurate to the nearest pixel, and therefore we use a threshold (which we call the *error threshold*) of one half of a pixel per visible trajectory point per frame.The details of the four steps listed above may be found in [7] or [10].

4 Experiments

Our algorithm was tested on two real image sequences of 50 frames: (1) the cylinder sequence, consisting of images of a cylinder rotating around a nearly vertical axis and a box moving right with respect to the cylinder and the background, and (2) the robot arm sequence, consisting of images of an Unimate® PUMA® Mark III robot arm with its second and third joints rotating in opposite directions. These sequences show the capabilities of the approach, and also demonstrate some inherent limitations of motion based segmentation and of monocular image sequence based motion estimation.

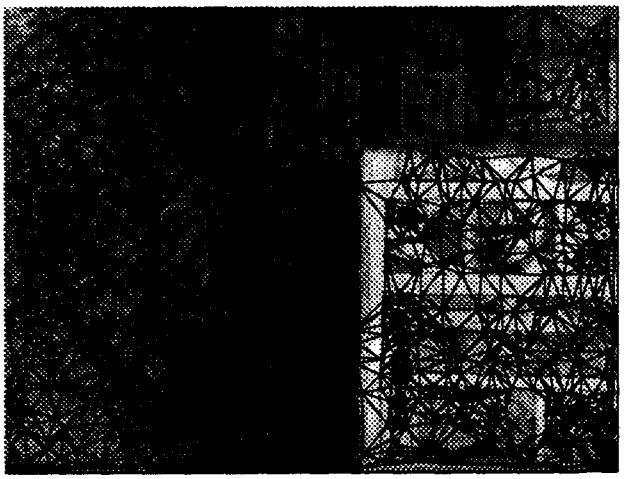


Figure 1 The image sequence segmentation found for the cylinder sequence (the segmentation is superimposed on the last frame of the sequence).

Table 1 Comparison of the parameters estimated by the algorithm and the true parameter for the cylinder image sequence experiment.

Parameters	Estimated	Actual
w	-0.022	-0.017
$\hat{\mathbf{F}}$	(0,0.94)	(0, 0.90)
$\hat{\mathbf{v}}$	(0.29, -0.19)	(0.14, 0)

Trajectories were detected using the algorithm described in [7] (using a method described in [11]), which found 2598 trajectories in the cylinder sequence and 202 trajectories in the robot arm sequence. These trajectories were input to the image sequence segmentation algorithm described in Section 3, which partitioned the trajectories into groups corresponding to different rigid objects and estimated the motion and structure parameters.

The segmentation for the cylinder sequence is shown in Figure 1. The algorithm separated out the three image regions: the cylinder, the box, and the background. The cylinder is rotating, and thus its structure can be recovered from the image sequence. Figure 1 shows a projection along the cylinder axis of the 3D point positions calculated from the 1456 points on the cylinder. The points lie very nearly on a cylindrical surface. Table 1 shows the estimated and the actual motion parameters for the cylinder. The error in the ω estimate is large because the cylinder is rotating around an axis nearly parallel to the image plane and, as pointed out in [12], a rotation about an axis parallel to the image plane is

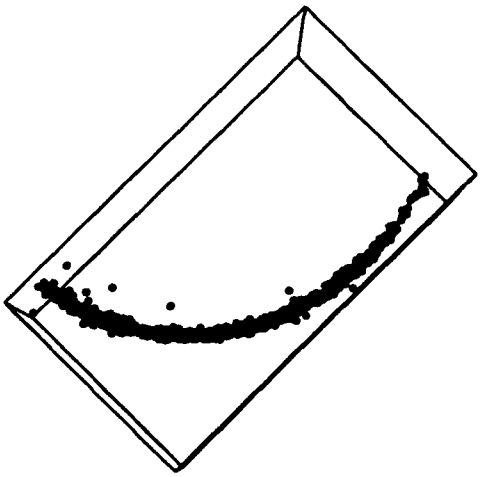


Figure 2 An end-on view of the three-dimensional point positions calculated by our structure and motion estimation algorithm from point trajectories derived from cylinder image sequence.

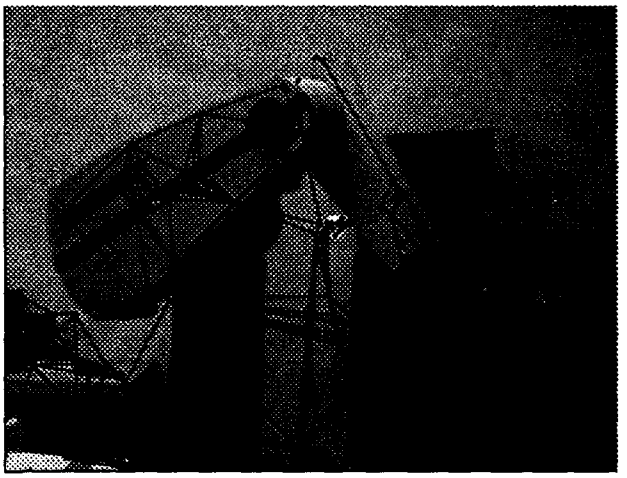


Figure 3 The image sequence segmentation found for the robot arm sequence (the segmentation is superimposed on the last frame of the sequence).

inherently difficult to distinguish from translation parallel to the image plane and perpendicular to the rotation axis (this also explains the error in $\hat{\mathbf{v}}$). Note that the predicted trajectory point positions still differ from the actual positions by an average of less than the error threshold of 0.5 pixel. The accuracy of the motion and structure estimation algorithm for less ambiguous motion is illustrated in the experiments on the robot arm sequence.

The image sequence segmentation for the robot arm sequence is shown in Figure 1. Note that several stationary feature points (only two visible in Figure 1) in the background are grouped with the second segment of the arm. This occurs because any stationary

Table 2 Comparison of the estimated and the true parameter values for the second (larger) segment of

Parameters	Estimated	Actual
w	0.0133	0.0131
$\hat{\mathbf{F}}$	(-0.51, -0.008)	(-0.46, 0.016)
$\hat{\mathbf{v}}$	(0.02, -0.07)	(0, 0)

Table 3 Comparison of the estimated and the true parameter values for the third (smaller) segment of the robot arm.

Parameters	Estimated	Actual
w	-0.0127	-0.0131
$\hat{\mathbf{F}}$	(-0.43, 0.045)	(-0.46, 0.016)

point lying on the projection of a rotation axis with no translational motion will fit the motion parameters of the rotating object. Thus these points are grouped incorrectly due to an inherent limitation of segmenting an image sequence on the basis of motion alone. The remaining points are grouped correctly into three image regions: the second and the third segments of the robot arm, and the background. The two robot arm segments are rotating and their three-dimensional structure was recovered by the motion and structure estimation algorithm.Only a small number of feature points were associated with the robot arm segments making it difficult to illustrate the structure on paper, but the estimated motion parameters of the second and third robot arm segments are shown in Table 2 and Table 3, respectively. Note that all the motion parameters were very accurately determined.

5 Conclusions

The main features of our method are: (1) motion and structure estimation and segmentation processes are integrated, (2) frames are processed sequentially with continual update of motion and structure estimates and segmentation, (3) the motion and structure estimation algorithm factors the trajectory data into separate motion and structure matrices, (4) aside from and SVD, the motion and structure estimation algorithm is closed form with no nonlinear iterative optimization required, (5) the motion and structure estimation algorithm provides a confidence measure for evaluating any particular segmentation,

References

[1] C. Debrunner and N. Ahuja, "A Direct Data Approximation Based Motion Estimation Algorithm," in *Proc. of the 10th ICPR*, Atlantic City, NJ, pp. 384-389, June 1990.

[2] C. Tomasi and T. Kanade, "Factoring Image Sequences into Shape and Motion", in *Proc. of the IEEE Motion Workshop*, Princeton NJ, pp. 21-28, October 7-9, 1991.

[3] T. Kanade, *personal communication*, October, 1991.

[4] G. Adiv, "Determining Three-Dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects," *IEEE Transactions on PAMI*, Vol. 7, pp. 384-401, July 1985.

[5] J. R. Bergen, P. J. Burt, R. Hingorani and S. Peleg, "Multiple Component Image Motion: Motion Estimation," in *Proc. of the 3^d ICCV*, Osaka, Japan, December 1990

[6] T. E. Boult and L. G. Brown, "Factorization-based Segmentation of Motions," in *Proc. of the IEEE Motion Workshop*, Princeton NJ, pp. 21-28, October 7-9, 1991.

[7] C. Debrunner, "Structure and Motion from Long Image Sequences," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, August 1990.

[8] C. Debrunner and N. Ahuja, "Estimation of Structure and Motion from Extended Point Trajectories," submitted.

[9] S. Y. Kung, K. S. Arun and D. V. B. Rao, "State-Space and Singular-Value Decomposition-Based Approximation Methods for the Harmonic Retrieval Problem," *J.of the Optical Society of America*, pp.1799-1811, December 1983.

[10]C. Debrunner and N. Ahuja, "Motion and Structure Factorization and Segmentation of Long Multiple Motion Image Sequences," submitted.

[11]S. D. Blostein and T. S. Huang, "Detecting Small, Moving Objects in Image Sequences using Sequential Hypothesis Testing," *IEEE Trans. on Signal Proc.*, Vol. 39, pp. 1611-1629, July 1991.

[12]J. Weng, T. S. Huang and N. Ahuja, "Motion and Structure from Two Perspective Views: Algorithms, Error Analysis, and Error Estimation," *IEEE Transactions on PAMI*, Vol. PAMI-11, pp. 451-476, May 1989.

Motion Analysis of Articulated Objects with Applications to Human Ambulatory Patterns

Richard J. Qian *Thomas S. Huang*

Beckman Institute
University of Illinois at Urbana-Champaign
405 North Mathews Avenue
Urbana, IL 61801

Abstract

Understanding the visual motion of articulated objects has broad applications. In this paper, we present new algorithms for the recovery of 3D structure from the motion of certain types of joints of articulated objects under the assumption of perspective projection. Then by the assumption that a human body can be considered as an articulated object and each part of the articulated object is rigid, we apply these algorithms to motion analysis of human ambulatory patterns in a combined way. The algorithms have been tested in terms of uniqueness of real solution and speed of convergence with synthesized data.

1. Introduction

An articulated object is a collection of rigid or non-rigid bodies which are connected by joints. Human vision systems have significant ability to understand the visual motion of articulated objects. The further study of how to understand the visual motion of articulated objects by means of computer vision may help us to gain insight into the nature of the way how biological visual systems work. Despite a large body of motion analysis works on purely rigid or even now purely nonrigid objects [Huang 87], we found that only a few published papers dealt with articulated objects which are neither purely rigid nor purely non-rigid objects. Hoffman and Flinchbaugh presented an algorithm for the recovery of 3D structure from biological motion in 1982. Because they aimed at computing the 3D structure and motion of animal limbs, a "planarity assumption" is introduced in their paper, i.e. the part(s) around a joint of an articulated object always move inside a single plane. Orthographic projection is also assumed. Their results are: 1) given three distinct orthographic projections of the two endpoints of a rigid rod which is constrainted to rotate in a plane, the structure and motion compatiable with the three views are uniquely determined; 2) given two distinct orthographic projections of the three endpoints of two rigid rods linked in a hinge joint to form a pairwise-rigid structure which is constrained to move in one plane, the structure and motion compatiable with the two views are uniquely determined. No experimental results are reported. Webb and Aggarwal presented their algorithm for the recovery of 3D structure from fixed-axis motion in 1981. Orthographic projection is also assumed in their paper. Their result is that the 3D structure of two points which execute fixed-axis rotation can be uniquely determined with four consequential image frames. Experimental results with real image data have been reported, but the accuracy of the results is considered as poor by the authors.

Understanding the visual motion of articulated objects has also broad applications. One application may be to apply the developed algorithms for 3D structure recovery from the motion of articulated objects to motion analysis of human ambulatory patterns. Studying how to understand the visual motion of human ambulatory patterns is of great importance for both vision research and applications. For example, it can be used in model-based image coding and transmission such that we may only need to transmit a few estimated motion parameters instead of sending a whole image sequence if certain human body models are available at both sender and receiver sides [Kimoto 91]. Another important application of understanding the visual motion of articulated objects is in robotic vision where most robot arms are ideal articulated objects.

Following this section, we will first develop

algorithms for the recovery of 3D structure from the motion of certain types of joints of articulated objects in Section 2. Then in Section 3, we will study how to apply these algorithms to motion analysis of human ambulatory patterns. Finally in Section 4, we will give a brief summary and discuss some future directions.

2. 3D Structure from Motion of Articulated Objects

It is noticed that the articulated objects discussed in both Hoffman and Webb's papers are pairwise-rigid, i.e. each part of the articulated object is rigid, and moreover each joint has at most one degree of rotational freedom. In this paper, we also emphysize such kind of articulated objects. However, we will study the cases under the assumption of perspective projection, because in practice camera projection is always perspective and in some cases using orthographic projection as an approximation cannot achieve the desired accuracy.

Systematically, we can write equations which describe the whole articulated object no matter how many joints it has [Featherstone 87]. Due to the high order of the nonlinear equations, however, it is very difficult to solve such equations and the uniqueness of the real solution of the equations is also hard to be studied. Alternatively, here we select a sequential way to solve the whole system. Only one joint is considered at a time. In the following subsections, we will first develop algorithms for the recovery of 3D structure from fixed-axis motion. Then we will discuss the more restricted cases, i.e. the cases of planar motion. For the sake of simplicity, we will use identical characters to denote points and their associated coordinate vectors in 3D space. For example, if there is a point p associated with a 3D coordinate vector of $(x, y, z)^T$, then we will still use character p to denote that vector, i.e. we let $p = (x, y, z)^T$. We will use $\overline{p_0p_1}$ to denote the vector from point p_0 to point p_1, i.e. $\overline{p_0p_1} = p_1 - p_0$, and use $\|\overline{p_0p_1}\|$ to denote its length. Because perspective projection is assumed, we have image coordinates and their corresponding 3D coordinates related as follows:

$$X = \frac{x}{z} \tag{2-1}$$

$$Y = \frac{y}{z} \tag{2-2}$$

where (X, Y) are image coordinates and (x, y, z) are the corresponding 3D coordinates, z is the depth. Then for any point p, we have:

$$p = (X, Y, 1)^T \cdot z \tag{2-3}$$

In the following sections, we will write equations in terms of the related 3D vectors. It is equivalent to write those equations in terms of images coordinates which are usually supposed to be known and 3D coordinates which are unknown. In fact, we can convert each equation from either of the two forms to the other by using formula (2-3).

2.1. Algorithm for 3D Structure from Fixed-Axis Motion (Algorithm 2.1)

Suppose we are given an image sequence of one part of a moving articulated object, see Figure 2.1-1. The only allowed motion is that rod $\overline{p_0p_1}$ can rotate around an unknown but fixed axis through joint p_0. The 3D coordinates (or equivalently the depth only) of the joint p_0 are given. The goal is to recover the 3D coordinates of points p_1 in 3D space from the given image coordinates.

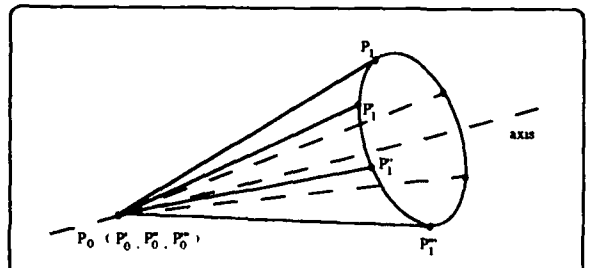


Figure 2.1-1

We attack this problem by solving a set of nonlinear equations. Using four consecutive frames, we can first write three degree-2 equations based on the rigidity constraint:

$$\|\overline{p_0p_1}\| = \|\overline{p_0'p_1'}\| = \|\overline{p_0''p_1''}\| = \|\overline{p_0'''p_1'''}\| \tag{2.1-1}$$

Because the rod rotates around a fixed axis through the joint, namely the end point p_1 remains in a single plane during the rotation. Then we have another degree-3 equation:

$$\overline{p_1''p_1'''} \cdot \overline{p_1p_1'} \times \overline{p_1'p_1''} = 0 \qquad (2.1\text{-}2)$$

Note we have four unknows. And we have set up four nonlinear equations. From Bezout's Theorem, the number of possible solutions is no larger than $2^3 \cdot 3 = 24$.

2.2. Algorithms for 3D Structure from Planar Motion

A. Case 1 (Algorithm 2.2-A)

Here again we are given an image sequence of one part of a moving articulated object, see Figure 2.2-A. But rod $\overline{p_0p_1}$ now is restricted to remain in a plane during its rotation. The 3D coordinates (or equivalently the depth only) of the joint p_0 are still given. The goal is to recover the 3D coordinates of points p_1 in 3D space from the given image coordinates.

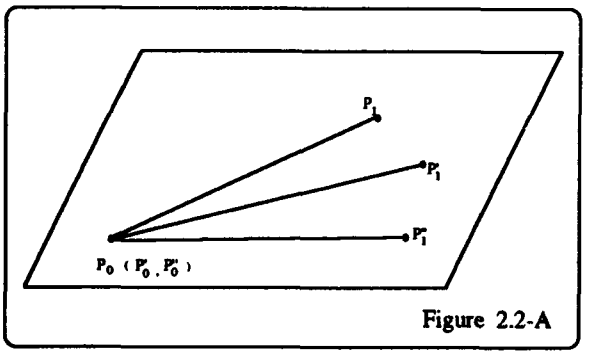


Figure 2.2-A

Using only three consecutive frames this time, we can write two degree-2 equations based on the rigidity constraint:

$$\|\overline{p_0p_1}\| = \|\overline{p_0'p_1'}\| = \|\overline{p_0''p_1''}\| \qquad (2.2\text{-}A1)$$

Then based on the planarity assumption, we can write another degree-3 equation:

$$\overline{p_0p_1} \cdot \overline{p_0'p_1'} \times \overline{p_0''p_1''} = 0 \qquad (2.2\text{-}A2)$$

In this case, we have three unknowns in total. Three nonlinear equations are available. From Bezout's Theorem, the number of solutions is no larger than $2^2 \cdot 3 = 12$.

B. Case 2 (Algorithm 2.2-B)

Fewer frames are needed for 3D structure recovery, if we consider two rods at a time, and if the motion is planar, i.e. the two rods remain in a single plane during their rotation. See Figure 2.2-B.

With only two consecutive frames, our first two degree-2 equations still come from the rigidity constraints:

$$\|\overline{p_0p_1}\| = \|\overline{p_0'p_1'}\| \qquad (2.2\text{-}B1)$$

$$\|\overline{p_1p_2}\| = \|\overline{p_1'p_2'}\| \qquad (2.2\text{-}B2)$$

Then based on the planarity assumption, we have two degree-3 equations:

$$\overline{p_1'p_2'} \cdot \overline{p_0p_1} \times \overline{p_1p_2} = 0 \qquad (2.2\text{-}B3)$$

$$\overline{p_1'p_0'} \cdot \overline{p_0p_1} \times \overline{p_1p_2} = 0 \qquad (2.2\text{-}B4)$$

There are four unknowns and four nonlinear equations are available. From Bezout's Theorem, the number of solutions is no larger than $2^2 \cdot 3^2 = 36$.

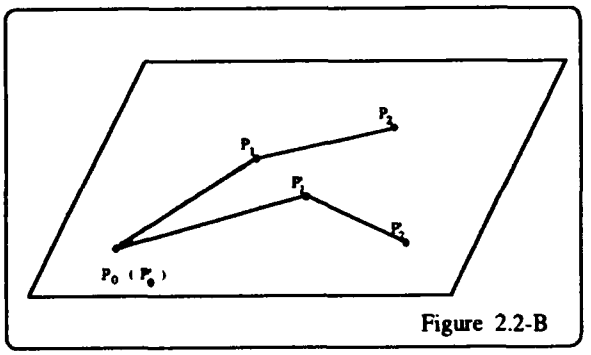


Figure 2.2-B

To analyze the uniqueness of real-valued solution of the equations in the above cases where perspective projection is assumed is much more difficult than the corresponding cases under the assumption of orthographic projection. At this point, we have only done a number of

numerical experiments. We solve the above nonlinear equations by using continuation method. We found that in all cases the real solution was unique up to a reflectance. The computing time is resonable. For example, it takes about five minutes on the average for the algorithm to converge for the equations in 2.2-B on a SUN-3 workstation.

3. Motion Analysis of Human Ambulatory Patterns

Up to now, few papers in computer vision have discussed motion analysis of human ambulatory patterns, still fewer have done it in a quantitative way. Johansson published the first related paper in 1973. Besides the sophisticated psycological experiments, only rough qualitative explanation is given in his paper [Johansson 73, 76]. Rashid's paper published in 1980 describes his work on the segmentation of moving light patterns which are produced by moving articulated objects like a human being or a dog against stationary background. The segmentation is done based on 2D information in images. No further 3D information recovery or motion estimation has been reported. In fact, the paper claims that the segmentation results can be a useful starting point for further analysis [Rashid 80]. O'Rourke and Badler's work in 1980 is model-based. Different from others, the proposed algorithm employs a complete human body model to track input human body motion. In the paper, only synthesized input data has been tested with the algorithm. Because the tracking results are heavily based on the segmentation results from images, it is not clear how well the algorithm will work with real images [O'Rourke 80]. Recently, Terzopolous and Metaxas suggest to use parameterized superqudrics to model human bodies and then do motion analysis based on that model. Unfortunately, no further details have been reported in their paper [Terzopolous 90].

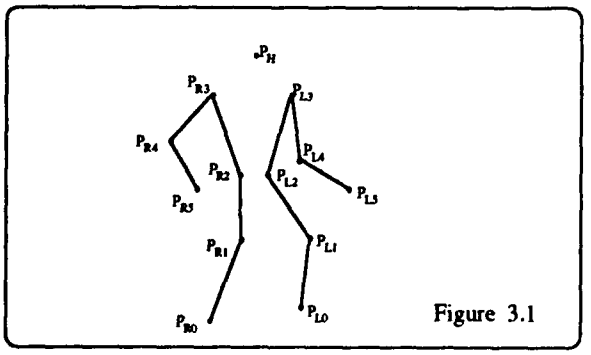


Figure 3.1

In this section, we apply the proposed algorithms for 3D structure recovery from the motion of articulated objects to motion analysis of human ambulatory patterns. We assume that the segmentation of human bodies and the extraction of the joints on human bodies from images have already been completed. In other words, we start with the obtained skeletons of human boides in image sequences. In each image frame, typically such a skeleton may look like the one in Figure 3-1.

Based on anatomical reasons, it is suggested that the four limb joints, i.e. p_{R1}, p_{L1}, p_{R4}, and p_{R4} in the figure, allow only planar motion during ambulatory motion [Hoffman 82]. And other joints allow only fixed-axis motion [Webb 81]. So, we can apply our algorithms developed in the previous section to them correspondingly. It is observed that a human body moves periodically during ambulatory motion. Each period consists of two halves corresponding to the right and the left, respectively. During each half of a period, one foot remains on the ground stationarily. So we can use that foot as a reference point when we want to do motion analysis during that time interval. Because we use image sequences from a single camera, we can only recover the relative depth information [Huang 87]. Hence, without loss of generality, we can assume that the 3D coordinates of the reference point are given. Then for a time instant t which belongs to a selected half of a period, we can start our analysis from bottom up as follows. We wili use $\bar{p}$ to denote a point p whose 3D coordinates are given or have been determined.

Step 1.

Algorithm 2.2-B $(\overline{p_{R0}(t)}, p_{R1}(t), p_{R2}(t))$ --> $(\overline{p_{R1}(t)}, \overline{p_{R2}(t)})$, *rotation* R_{R012};
transform p_{R2}, p_{R3}, p_{R4}, p_{R5}, p_{L2}, p_{L1}, p_{L0}, p_{L3}, p_{L4}, and p_{L5} *based on* R_{R012};

Step 2.

Algorithm 2.1 $(\overline{p_{R2}(t)}, p_{R3}(t))$ --> $(\overline{p_{R3}(t)})$, *rotation* R_{R23};
transform $\overline{p_{R3}}$, p_{R4}, and p_{R5} *based on* R_{23};
Algorithm 2.1 $(\overline{p_{R2}(t)}, p_{L2}(t))$ --> $(\overline{p_{L2}(t)})$, *rotation* R_{RL};
transform $\overline{p_{L2}}$, p_{L1}, p_{L0}, p_{L3}, p_{L4}, and p_{L5} *based on* R_{22};

Step 3.

Algorithm 2.2-B $(\overline{p_{R3}(t)}, p_{R4}(t), p_{R5}(t))$ --> $(\overline{p_{R4}(t)}, \overline{p_{R5}(t)})$;
Algorithm 2.2-B $(\overline{p_{L2}(t)}, p_{L1}(t), p_{L0}(t))$ -->

$(\overline{p_{L1}}(t), \overline{p_{L0}}(t))$;

Algorithm 2.1 $(\overline{p_{L2}}(t), p_{L3}(t))$ --> $(\overline{p_{L3}}(t))$, *rotation* R_{L23};;

transform $\overline{p_{L3}}$, p_{L4}, and p_{L5} *based on* R_{L23};

Step 4.

Algorithm 2.2-B $(\overline{p_{L3}}(t), p_{L4}(t), p_{L5}(t))$ --> $(\overline{p_{L4}}(t), \overline{p_{L5}}(t))$;

It is obvious that the way to combine the algorithms to recover the 3D information of all the joints is not unique. For example, we can employ *Algorithm*2.2–*A* instead of *Algorithm*2.2–*B*. But as we pointed out in Section 2.2, the former algorithm uses more frames than the latter.

4. Summary

In this paper, we have developed three algorithms under the assumption of perspective projection for the recovery of 3D structure from the motion of several types of joints of articulated objects, viz. joints which allow only fixed-axis rotation and joints which allow only planar rotation. Then we apply these algorithms to analysis of human ambulatory motion in a combined way. Our numerical experimental results with synthesized data have indicated that the real solution from the proposed algorithm was always unique up to a reflectance and the time of convergence was reasonable.

Understand the visual motion of human body is an important issue which has been started quite recently. There is a long way to go and many sub-topics now are totally open. Even in the special case of understanding human ambulatory motion, there are still a number of topics which need to be studied. Among them, the segmentation of human bodies from images is the first and also maybe the hardest. Gait analysis is another important and also interesting topic to be studied. And in some applications, more sophisticated human models and/or motion models may be needed. In the meantime, testing the present algorithms with real image data and finding ways to improve them are our near-term research goals.

Acknowledgement This work was supported by National Science Foundation Grant IRI-89-02728 and IRI-89-08255.

References

Featherstone, R. [1987] "Robot Dynamics Algorithms", *Robotics: Vision, Manipulation and Sensors*, ed. by Kanade,T., Kluwer Academic Publishers, 1987.

Huang, T.S. [1987] "Motion Analysis", *Encyclopedia of Artificial Intelligence*, ed. by Shapiro, S., Wiley and Sons, 1987. pp.620-632.

Hoffman, D.D. and Flinchbaugh, B.E. [1982] "Interpretation of Biological Motion", *Biological Cybernetics*, vol.42, 1982. pp.195-204.

Johansson, G. [1973] "Spatial-temporal differentiation and integration in visual motion perception", *Psychology Research*, vol.38, 1973. pp.379-393.

Johansson, G. [1976] "Visual Motion Perception", *Scientific American*, vol.232, June, 1976. pp.76-88.

Kimoto, T. and Yasuda, Y. [1991] "Hierachical Representation of the Motion of a Walker and Motion Reconstruction for Model-Based Image Coding", *Optical Engineering*, vol.30, no.7, 1982. pp.888-903.

Kozlowski, L.T. and Cutting, J.E. [1977] "Recognizing the Sex of a Walker from Dynamic Point-light Display", *Perception and Psychophysics*, vol.21, no.6, 1977. pp.575-580.

O'Rourke, J. and Badler, N.I. [1980] "Model-based Image Analysis of Human Motion Using Constraint Propagation", *PAMI*, vol.2, no.6, 1980. pp.523-536.

Rashid, R.F. [1980] "Towards a System for the Interpretation of Moving Light Displays", *PAMI*, vol.2, no.6, 1980. pp.574-581.

Terzopoulos, D. and Metaxas, D. [1990] "Dynamic 3D Models with Local and Global Deformation: Deformable Superquadrics" , *ICCV'90*, 1990. pp.606-615.

Webb, J.A. and Aggarwal J.K. [1981] "Structure from Motion of Rigid and Jointed Objects", *IJCAI'81*, 1981. pp.686-691.

Temporal Texture Analysis

Ramprasad Polana and Randal C.Nelson *
Department of Computer Science
University of Rochester
Rochester, New York 14627

Abstract

We describe a method of visual motion recognition applicable to a range of naturally occurring motions that are characterized by spatial and temporal uniformity. The underlying motivation is the observation that, for objects that typically move, it is frequently easier to identify them when they are moving than when they are stationary. Specifically, we show that certain statistical spatial and temporal features that can be derived from from approximations to the motion field have invariant properties, and can be used to classify regional activities such as windblown trees, ripples on water, or chaotic fluid flow, that are characterized by complex, non-rigid motion. We refer to the technique as *temporal texture analysis* in analogy to the techniques developed to classify gray-scale textures. This *recognition* approach contrasts with the *reconstructive* approach that has typified most prior work on motion. We demonstrate the technique on a number of real-world image sequences containing complex movement. The work has practical application in monitoring and surveillance, and as a component of a sophisticated visual system.

1 Motion Recognition

The use of visual motion for the quantitative reconstruction of world geometry has been extensively studied. This has sometimes obscured the fact that motion can also be used for recognition. In fact, in biological systems, the use of motion information for recognition is often more evident than its use in reconstruction. A simple example occurs in the case of the common toad *Bufo bufo* for which any elongated object within a certain size range that exhibits motion along the long axis is identified as a potential food item, and elicits an orienting response [Ewart, 1987]. Birds ignore the natural movement of trees in the wind, but respond immediately to the approach of a predator. More generally, stylized movements seem to be a universal form of communication between animals with eyes, from the aggressive posturing of various fiddler crabs (*Uca* species), to the mating dance of the blue footed booby (*Sula Nebouxi*), to the expressive facial movements of baboons.

Humans have a remarkable ability to recognize different kinds of motion, both of discrete objects, such as animals or people, and in distributed patterns as in windblown leaves or waves on a pond. A classic illustration of motion recognition by humans is provided by Moving Light Display experiments where the sole source of information about a moving actor is provided by lighted points attached to a few joints [Johansson, 1973]. People shown these images dismiss single frames as meaningless dot patterns but can recognize characteristic gaits such as running or walking, and even gender and familiar individuals from the sequential presentation.

Such abilities suggest that, in the case of machine vision, it might be possible to use motion directly as a means of recognition rather than indirectly through a geometric reconstruction. In addition to the biological motivations, there are computational reasons for considering motion as a recognition modality. One advantage is that the motion field, insomuch as it can be extracted at all, is robust with respect to lighting changes, and much more simply related to shape than is image luminance. Furthermore, if the task is to find an object that is known to be moving, motion can be used to efficiently presegment the scene into regions of high and low interest. This can frequently be done even if the observer is itself moving [Nelson, 1991]

Motion recognition has seen limited attention in the literature Most of the work that has been done involves the analysis of moving light displays. A domain independent approach to this problem is given by Rashid [1980]. Goddard [1988] considers the representation and recognition of event sequences from moving light display images of human beings. He uses the joint angles and angular velocities computed from the motion of the dots in the light displays. The joint angles and angular velocities are invariant to scale, translation, and rotation in the image plane. A challenging part in computing these invariants is to recover the underlying connectivity of the individual dots in the MLD images.

A number of potential applications for motion recognition exist. One example is detecting malfunctions in industrial automation processes. This could be done both by detecting unusual motion or by noting the absence

*This work is supported by contracts NSF IRI-9010692 and AFOSR 91-0288.

of a familiar motion sequence that should be present in normal operation. In some applications, a visual system could monitor a process more effectively, and be retrofitted more easily than a large set of specialized sensors. Motion detection can also be used in surveillance applications where the system must distinguish environmental motions such as plants waving in the wind from unfamiliar motions such as the movement of an intruder into prohibited areas. Other examples include gesture recognition [Rhyne and Wolf, 1986] and handwriting analysis.

2 Temporal Texture

Classical gray-level texture analysis is concerned with the identification of spatial invariances in the gray-level patterns in an image region. These invariances may be either structurally or statistically defined. The basic idea is to characterize different sorts of "stuff" of indeterminate spatial extent in terms of such invariances. In this article we extend this basic idea into the temporal dimension with the idea of recognizing similar "stuff" in dynamic scenes. This is motivated in part by the existence of a large class of natural phenomena that seem to have characteristic motions, but indeterminate spatial extent. Examples include windblown trees or grass, turbulent flow in cloud patterns, ripples on water, falling snow, the motion of a flock of birds or a crowd of people, etc. The motion in a temporal texture is distinct from that in patterns such as walking, cycling etc. which involve structure at a single location.

Temporal texture could be analyzed directly as a three dimensional signal using generalizations of the techniques applied to two-dimensional fields. However, since most changes along the time dimension are due to motion in the image, it makes sense to preprocess the time-varying image to obtain motion information, as it is in object motion that the physical invariances lie. In this case, a natural choice is the optic flow field. The basic source of information is thus a time varying vector field representing an approximation to the two dimensional motion field induced by movement in the world. Such a field contains considerably more information than the scaler valued field associated with gray level texture analysis. In addition, the direction and magnitude of motion have a more direct relationship to typically salient events in the world than the gray-level of a single pixel. Consequently, certain types of recognition might be expected to be easier. One problem with using optic flow is that it is difficult compute accurately. One solution is to devise measures that are insensitive to inaccuracy. Another is to utilize partial information. An example is the gradient parallel component of the optic flow, which is simpler to compute locally from an image sequences than the full motion field.

Despite the differences in domain, some techniques of spatial texture analysis are applicable to temporal textures. Spatial texture analysis is traditionally performed using either statistical or syntactic methods. Statistical methods utilize measures of local features that are expected to be similar within patches of the same texture. Examples include gray-level co-occurrence matrices, gradient uniformity measures, local Fourier power spectra, average response of oriented masks, and estimates of parameters for Markov generating processes. Syntactic approaches are most appropriate for highly regular textures and involve analyzing the geometric arrangement of primitive structural elements. In the case of natural temporal textures, techniques similar to the statistical gray-level methods seem most appropriate, and most of the features described in this article are of this type. As with spatial textures, the main criteria for selecting features are that they change little within a given texture (i.e. an area of the same stuff), and that they vary significantly between different textures.

The dimensionality of the vector-valued flow field and the fact that measures can be made in both space and time allows considerable latitude in designing features. Since textures are characterized by statistical regularities in the occurrence of local structure, extraction of features useful for classification generally involves at least two tiers of processing: A local feature extraction stage, and (at least one) spatially or temporally extended integration stage. Local features can be any useful quantity that can be associated with a point in the image. Examples include flow magnitude and direction, differential measures such as divergence and curl, and local uniformity measures. The spatio-temporal motion energy filters introduced by Heeger [1987] could also provide useful measures in this context. Typically these are expected to vary within a texture, thus necessitating the integration phase. Extended measures are most frequently based on quantities such as means or variances, but other extended measures, such as Fourier coefficients and co-occurrence statistics can be used. The most typical structure for a temporal texture feature involves extended spatial or temporal (or both) measures of spatio-temporal microfeatures. Features can also be derived from extended spatial measures of extended temporal features and vice versa.

In order to simplify the motion preprocessing, we considered features based on the gradient parallel component of the motion field, also referred to as the normal flow. The simplest local motion measures are the magnitude and direction of the normal flow. We examine several statistical features based on the distribution of these first-order quantities. The direction and magnitude can be combined locally, both spatially and temporally to obtain second order local motion measures. We also examine features based on the distribution of some second order measures. All these are described below.

A useful first order statistic can be derived from the distribution of flow directions. Intuitively, what is being measured is the non-uniformity in direction of motion. Our non-uniformity statistic was computed by discretizing the direction into 8 possible values, computing a histogram over the relevant neighborhood of the image, and summing the absolute deviation from a uniform distribution. It should be noted that the normal flow direction at a pixel is parallel (or anti-parallel) to the gradient direction. Thus measures based on the normal flow direction alone depend on the underlying intensity texture. To reduce this dependence, the normal flow directions in

the histogram are normalized by the 4-way histogram of gradient directions. This feature is invariant under translation, rotation, and temporal and spatial scaling.

A useful statistic based on the distribution of the normal flow magnitude is the average flow magnitude divided by its standard deviation. The scaling by the standard deviation has the effect of making the measure robust under scaling changes. One way to think of this statistic is as a measure of "peakiness" in the velocity distribution. It is also invariant under translation, rotation, and temporal and spatial scaling.

Second order statistics of the normal flow direction distribution can be derived from the difference statistics, which give the number of pixels pairs at a given offset which differ in their values by a given amount. These difference statistics can be represented by a co-occurrence matrix of the normal flow direction surrounding a pixel. Co-occurrence matrices are computed for four directions, (horizontal, vertical, positive diagonal and negative diagonal), at a distance proportional to the average flow magnitude. This yields invariance with respect to scaling. In each direction the ratio of the number of pixel pairs differing in direction by at most one to the number of pixel pairs differing by more than one is computed. This ratio is the sum of the first two difference statistics to the sum of the last three difference statistics. Logarithms of the resulting ratios are used as a feature in each of the four directions, and represent a measure of the spatial homogeneity of the flow. These features are invariant under translation, rotation and scaling.

Finally, we considered statistics of some second order flow features, namely, estimates of the divergence and curl of the motion field obtained from the normal flow. Positive and negative divergence, and positive and negative curl were taken as separate features to give four different second order features. The features used are the mean values of these quantities over the region of interest. They are invariant with respect to rotation and translation, but not scaling. If scale invariant features are desired, ratios of the differential measures can be used.

3 Experimental Results

A set of image sequences representing both oriented temporal textures such as flowing water and non-oriented textures such as leaves fluttering in the wind was digitized. In addition, sequences representing uniform expansion and rotation of a textured scene were obtained. These were used in classification experiments utilizing the features described above. Seven different texture samples, listed below, were used for the experiments.

A: uniformly expanding image produced by observer motion

B: fluttering of vertical paper bands

C: cloth waving in the wind

D: motion of tree leaves in a breeze

E: uniformly rotating image produced by observer roll

F: turbulent motion of water

G: flow of water in a river.

Representative examples are illustrated in Figure 1.

For each sample texture, two image sequences consisting of 16 256x256 pixel frames taken at 30 Hertz were split into quadrants to obtain eight independent sample image sequences of 128x128 pixels. The normal flow field was computed between each consecutive pair of image frames using a multi-resolution flow computation, with the direction of normal flow quantized to one of eight directions. The end result of the processing was a sample of 8 normal flow sequences of fifteen frames each for each texture.

Classification experiments were run using a nearest centroid classifier. More elaborate classifiers could be used, but the nearest centroid method gives a fairly direct indication of the utility of the features. The features used were those described in the previous section, namely

a: nonuniformity of normal flow direction

b: mean flow magnitude divided by its standard deviation

c: directional difference statistics in four directions

d: positive and negative curl and divergence estimates

The first four samples of each texture are used as a training set to compute the centroid of the cluster corresponding to that texture in the feature space. The different feature values are converted into common units by mapping the average of the resulting centroids to a unit vector. Table 1 contains the values of these features for each flow sample. It can be seen that, as desired, the within sample variation is small and the between sample variation is high. No single feature is sufficient to distinguish all the textures, but for each texture, there is at least one feature that clearly separates it from the others. For example, as would be expected, texture A containing an approaching object is distinguished by high divergence. For texture B, containing moving vertical bands, the second order difference feature in the vertical direction clearly separates it from the rest.

The remaining four samples are tested using a nearest centroid classification scheme. The results of classification are summarized in Table 2. Note that none of the features alone is sufficient to separate all the textures, but the combination gives one hundred percent success in the classification of the test cases. In fact, the second order features alone are sufficient in this case for successful classification in all cases.

Feature Combination	*Correct Classification*	*Percent Success*
all	28	100
c,d	28	100
b,d	24	85
a,c	21	75
d	21	75
c	20	71

Table 2: nearest centroid classification

texture	*b*	*c*				*a*	*d*			
	mag	*pos div*	*neg div*	*pos curl*	*neg curl*	*dir*	*hor*	*vert*	*pos diag*	*neg diag*
	1.0998	0.4626	-1.0012	0.2683	-0.2317	0.9473	2.1750	3.1676	2.6613	2.2411
A	1.0768	0.3979	-0.9544	0.2669	-0.2069	0.9224	2.6683	3.3278	3.7917	2.7856
approach	1.0282	0.3363	-0.9426	0.2480	-0.1867	0.9225	2.3663	3.1734	3.2725	2.4903
	1.0185	0.4229	-1.0183	0.3317	-0.2576	0.9185	2.5973	3.4585	3.3759	2.6836
	1.0836	0.5917	-0.6987	0.2758	-0.1991	0.4886	5.0131	9.0840	5.1928	5.1780
B	1.0091	0.6402	-0.5706	0.2404	-0.2236	0.6639	4.6796	8.3563	4.9623	5.0555
bands	1.0819	0.4677	-0.6251	0.2123	-0.2097	0.8374	4.3582	7.8780	4.5181	4.4092
	1.2212	0.5449	-0.5481	0.1887	-0.2031	0.6941	5.3194	8.9544	5.5406	5.4524
	1.4172	0.6485	-0.6200	0.3148	-0.3222	0.9282	4.2657	6.1709	4.8066	4.2894
C	1.5290	0.5305	-0.5570	0.3239	-0.3357	0.9171	4.9394	5.6819	6.1495	4.8262
cloth	1.2823	0.6102	-0.5976	0.3172	-0.3086	0.9425	3.3903	4.9728	3.8821	3.3389
	1.3934	0.6103	-0.6471	0.3373	-0.3425	0.9023	3.5964	4.7320	4.2765	3.5633
	0.9649	0.7083	-0.2160	0.1967	-0.2977	0.9473	1.4811	2.1032	2.2761	1.4669
D	1.0649	0.3060	-0.4348	0.2635	-0.1767	0.9521	1.5560	2.2878	2.3531	1.5746
leaves	0.8824	0.5276	-0.4368	0.2580	-0.2795	0.9687	1.2625	1.8688	2.0555	1.2391
	0.9512	0.3861	-0.3924	0.2940	-0.2644	0.9706	1.3002	1.8715	2.0534	1.2431
	1.1829	0.4379	-0.3957	0.0958	-0.5849	0.9296	2.9525	4.0765	3.5235	3.0259
E	1.0240	0.6216	-0.4206	0.0835	-0.6634	0.9422	3.2576	4.1850	4.0779	3.3940
rotate	1.0321	0.3829	-0.3530	0.0533	-0.6601	0.9356	2.9231	3.9704	3.6276	3.0769
	1.0877	0.5283	-0.3370	0.1106	-0.7256	0.9438	2.7883	3.7829	3.5971	2.9063
	1.1234	0.7287	-0.6374	0.4007	-0.3996	0.9464	2.4546	2.9726	3.9620	2.5215
F	1.2061	0.8110	-0.5876	0.3767	-0.4088	0.9295	2.6160	3.0522	4.3030	2.6992
turbulence	1.1065	0.5954	-0.7693	0.4223	-0.3971	0.9455	2.1867	2.7337	3.6717	2.2501
	1.0628	0.7998	-0.5266	0.4301	-0.4271	0.9455	2.1649	2.6112	3.6775	2.1524
	1.2939	0.4466	-0.5504	0.1912	-0.1619	0.8648	4.6374	5.1481	7.1544	4.7927
G	1.4944	0.4860	-0.3824	0.1717	-0.1876	0.8142	5.0256	5.6174	7.0384	5.1103
water	1.2587	0.5171	-0.5853	0.2065	-0.1867	0.8857	4.2978	4.7771	6.2181	4.5055
	1.5123	0.4486	-0.5282	0.2221	-0.2259	0.8870	3.8690	4.1761	6.0731	3.8762

Table 1: sample features

comp.	*b*	*c*				*a*	*d*				*eigenvalue*
1	0.292	0.515	0.955	0.015	-0.291	-0.201	0.154	0.205	0.178	0.160	54.954
2	-0.248	0.420	0.376	0.783	0.075	0.100	-0.822	-0.371	-0.894	-0.691	5.822
3	-0.089	0.783	-0.504	0.118	-0.159	0.007	0.157	0.058	-0.000	0.133	2.540

Table 3: first three principal components

We also performed a principal component analysis of these features to gauge the relative importance of different features in producing the variation in the sample values. The first three principal components of the entire data set are shown in Table 3. Note that the first principle component has a high eigenvalue, and relatively high proportions of the second order features, particularly positive and negative divergence. This is consistant with the finding that the second order features alone are sufficient for classification in this case. The principal components within each sample contain small absolute coefficients for the same second order features, showing that these features are most useful in classification.

4 Conclusion

We have described a method of motion recognition using temporal textures. This technique uses statistical measures of local motion features as components of a feature vector that can be used in standard classification methods. We identified several motion features that appear to have desirable properties for recognition, and illustrated their utility in classifying a sample of real-world temporal textures. Future work includes the analysis of other feature classes, including purely temporal features of the flow as well as Fourier techniques. We also plan to pursue the connection to qualitative analysis of motion, and to see whether such qualitative interpretation can allow the technique to be extended to the analysis of more structured movements such as those exhibited by individual people or animals.

References

[Ewart, 1987] J.P. Ewart. Neuroethology of releasing mechanisms: Prey-catching in toads. *Behavioral and Brian Sciences*, 10:337–405, 1987.

[Goddard, 1988] N.H. Goddard. Representing and recognizing event sequences. In *Proc. AAAI Workshop*

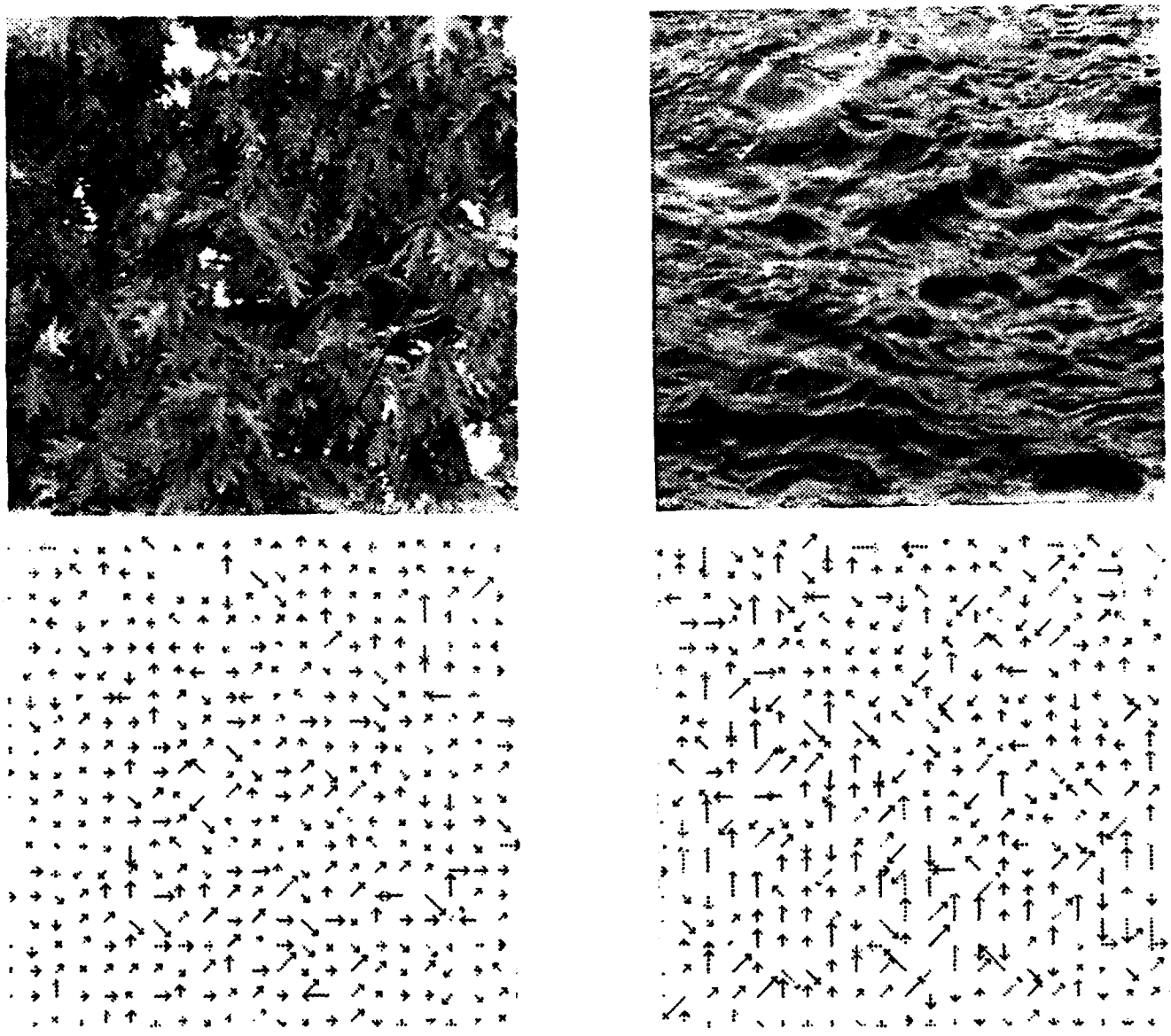


Figure 1: sample texture and normal flows of leaves and turbulent water

on Neural Architectures for Computer Vision, 1988.

[Heeger, 1987] D.J. Heeger. Optical flow from spatiotemporal filters. *IEEE*, pages 181–190, 1987.

[Johansson, 1973] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14:201–211, 1973.

[Nelson, 1991] R.C. Nelson. Qualitative motion detection by a moving observer. *International Journal of Computer Vision*, 7, 1991.

[Rashid, 1980] R.F. Rashid. *LIGHTS: A System for Interpretation of Moving Light Displays.* PhD thesis, Computer Science Dept, University of Rochester, 1980.

[Rhyne and Wolf, 1986] J.R. Rhyne and C.G. Wolf. Gestural interfaces for information processing applications. Technical Report 12179, IBM Research Report, 1986.

Subsection D:

Shape Recovery and Analysis

Direct Method For Reconstructing Shape From Shading

Paul Dupuis‡
Box F, Division of Applied Mathematics
Brown University
Providence, Rhode Island 02912

John Oliensis†
Department of Computer Science
University of Massachusetts at Amherst
Amherst, Massachusetts 01003 *

Abstract

A new approach to shape from shading is described, based on a connection with a calculus of variations/optimal control problem. The approach leads naturally to an algorithm for shape reconstruction that is simple, fast, provably convergent, and, in many cases, provably convergent to the correct solution. In contrast with standard variational algorithms, it does not require regularization. An explicit representation is given for the surface; uniqueness of the reconstruction (under suitable conditions) is an immediate consequence. Given a continuous image, the algorithm can be proven to converge to the continuous surface solution as the image sampling frequency is taken to infinity. Experimental results are presented for synthetic and real images, for general lighting direction.

1 Introduction

Shape from shading has traditionally been considered an ill–posed problem, with potentially infinitely many different surfaces corresponding to a shaded image. Therefore, most algorithms for reconstructing shape have incorporated *regularization* techniques to guarantee recovery of a unique, 'physically reasonable' surface solution.

More recently, it was suggested that shape from shading need not be ill–posed when the image contains singular points, i.e., maximally bright image points [1,4,16,17,14,13]. This was shown for the case of illumination from—or symmetric around—the camera direction in [14]. In addition, a general shaded image was shown to *uniquely* determine shape under the assumed lighting conditions [14]. Singular points provided the essential constraints.

Singular points continue to give strong constraints on the surface solutions for illumination from a general direction [13]. Thus, shape from shading should not be assumed ill–posed in general, and regularization should be used with caution. Also, the image of the occluding boundary gives no useful constraint on surface reconstruction [13]. Singular points, therefore, provide the primary constraints.

*This work was supported by the National Science Foundation under grants IRI-9113690, CDA-8922572†and NSF-DMS-8902333‡, and by a grant from DARPA, via TACOM, contract number DAAE07-91-C-R035†.

Nevertheless, shape–from–shading algorithms in the past have not taken full advantage of the strong constraints due to singular points. Algorithms based on the method of characteristic strips [3] have used these constraints explicitly, but in an approximate way. These algorithms have usually been applied to rather simple images, and are nonrobust in the presence of noise.

Most recent algorithms for recovering shape from shading have been based on the variational approach (e.g., [6,5,4]). These algorithms have had significant successes on complex images, but do not explicitly use the singular point constraints. This is seen experimentally in the fact that these algorithms do better on images with many singular points than on images with just one (see below and also [9]); yet for such simple images, the sole singular point is known to directly and uniquely constrain the surface reconstruction [1,17].

In this paper, an algorithm is presented that takes full advantage of the singular point constraints. It is simple, fast, provably convergent, and, in many cases, provably convergent to the correct solution. In particular, if the surface is known to be *unimodal* at a singular point in the image (i.e., locally concave or convex at this point), then the algorithm provably reconstructs the correct surface in a region around the singular point. The algorithm is robust against noise and, unlike previous algorithms, does not employ regularization. There is no problem with false minima, in contrast to the standard variational approach. Finally, this approach is capable of dealing with some orientation discontinuities—images for which the intensity function is only piecewise continuous.

The algorithm is based on establishing the equivalence of shape from shading to a calculus of variations/optimal control problem. This can be done for illumination from the camera direction, or for regions near unimodal singular points. For the general case with illumination from an arbitrary direction, the optimal control problem must be extended to a differential game. This equivalence facilitates the theoretical analysis of shape from shading, and makes the algorithm highly adaptable. It also gives intuition about the convergence performance of the algorithm. Below, we present a simple uniqueness proof for shape from shading which generalizes from the local uniqueness results of Bruss [1] and Saxberg [17]. This is possible because, in the optimal control representation, an expression for the surface corresponding to a shaded

image can be exhibited explicitly. Some of the results presented in this paper have also been derived in [18].

2 Shape from Shading as a Problem of Optimal Control

The imaged surface is assumed to be Lambertian, and viewed from above along the $-\hat{z}$ direction. It is represented in the explicit form $z(x,y)$, where $z : \mathbf{R}^2 \rightarrow \mathbf{R}$ is the *height function* to be reconstructed. We consider first the simpler case of illumination along the viewing direction $-\hat{z}$ (vertical light). The case of illumination from a general direction is discussed later.

Under these conditions, the image irradiance equation is:

$$I(x,y) = \frac{1}{(1+|\nabla z(x,y)|^2)^{1/2}}. \tag{2.1}$$

It is convenient to rewrite this in the eikonal form:

$$|\nabla z(x,y)|^2 = \frac{1}{(I(x,y))^2} - 1 \equiv V(x,y), \tag{2.2}$$

where $I(x,y) \in (0,\ 1]$, $V(x,y) \in [0,\ \infty)$. This type of equation arises frequently in the dynamical programming approach to problems of optimal control. In this section, we give a heuristic and indirect demonstration that it is equivalent to a calculus of variations/optimal control problem. Later, a direct and rigorous derivation will be given. Other possibilities (e.g. inclusion of a terminal cost) are also of interest for shape from shading, and will be discussed below.

Consider the following control problem: a 'particle' initially located at $(x_0,\ y_0)$ moves in the image plane in response to control parameters u, v, according to:

$$\dot{x} = u, \qquad \dot{y} = v, \qquad x(0) = x_0, \qquad y(0) = y_0. \tag{2.3}$$

The control parameters are to be chosen to minimize a cost function for the particle's trajectory $(x(s),\ y(s))$:

$$U(x_0,y_0,T) = \inf\{ \\ \tfrac{1}{2}\int_0^T ds\,\left(u(s)^2 + v(s)^2 + V(x(s),y(s))\right)\}. \tag{2.4}$$

Other control problems could also be considered [18]. In this equation, the minimal cost has been defined as a function of the trajectory starting point $(x_0,\ y_0)$. The infimization is over all piecewise continuous functions $u(\cdot)$, $v(\cdot)$ on $[0,\ T]$. Let

$$U(x,y) = \lim_{T\rightarrow\infty} U(x,y,T).$$

$U(\cdot)$ will turn out (in the unimodal case) to be the surface $z(x,y)$ up to a translation. To show this formally, we assume that $U(\cdot)$ is a differentiable function of the starting point, and formally demonstrate using dynamical programming that it satisfies eq. 2.2.

Let δT be a small time increment. Then:

$$U(x_0,y_0,T) = \inf_{(u,v)}\{ \\ U(x(\delta T),y(\delta T),T-\delta T) \\ +\tfrac{1}{2}\int_0^{\delta T} ds\,\left(u(s)^2 + v(s)^2 + V(x(s),y(s))\right)\}. \tag{2.5}$$

The explicit infimization is now over the part of the trajectory with $T \in [0,\ \delta T]$; the infimization over the rest of the trajectory is included in the cost function $U(\cdot, T-\delta T)$. Since δT is small, U can be expanded to first order in this quantity, which gives (for $\delta T \rightarrow 0$):

$$\frac{\partial U}{\partial T}(x,y,T) = \inf_{(u(0),v(0))}\{ \\ \tfrac{1}{2}\left(u^2(0) + v^2(0) + V(x,y)\right) \\ +\frac{\partial U}{\partial x}(x,y,T)u(0) + \frac{\partial U}{\partial y}(x,y,T)v(0)\}. \tag{2.6}$$

Performing the minimization over $u(0)$ and $v(0)$ yields:

$$u(0) = -\frac{\partial U}{\partial x}(x,y,T), \qquad v(0) = -\frac{\partial U}{\partial y}(x,y,T), \tag{2.7}$$

and

$$\frac{\partial U}{\partial T}(x,y,T) = \\ \tfrac{1}{2}\left[V(x,y)) - \left(\frac{\partial U}{\partial x}(x,y,T)\right)^2 - \left(\frac{\partial U}{\partial y}(x,y,T)\right)^2\right]. \tag{2.8}$$

Suppose that the image region under consideration is a small neighborhood of a singular point, at which $I = 1$ and $V = 0$. A minimal cost trajectory clearly moves toward regions of smaller V, and will converge to the singular point at which the incremental cost is zero. As the trajectory converges to this point, the total cost along the trajectory converges to a *finite* value. Therefore, the integration limit T in eq. 2.4 can be taken to infinity, and $U(x,y)$ is well defined. Since the time derivative vanishes, $U(x,y)$ satisfies:

$$\left(\frac{\partial U(x,y)}{\partial x}\right)^2 + \left(\frac{\partial U(x,y)}{\partial y}\right)^2 = V(x,y). \tag{2.9}$$

Since this is just the image irradiance equation, eq. 2.2, *$U(\cdot)$ can be identified with $z(x,y)$*. Also, u and v can be identified with $-p$ and $-q$, respectively, from eq. 2.7. Thus the minimal cost trajectories are curves of steepest descent, and are just the *characteristic strips* [3,13].

Note also that $U \geq 0$, and that $U = 0$ only at the singular point. Thus, the solution that is locally concave at the singular point has been automatically selected by this formulation; the solution locally convex at the singular point is just its negative. The solution U is unique, since it is equal to the infimum of the cost, which must be unique. Since an infimum of the cost always exists, the solution U always exists. It must be continuous, but need not be differentiable.

This formulation also gives a way of computing U. Clearly, $U(x,y,0) = 0$ for all $(x,\ y)$, while $U(x,y,T)$ is monotonically increasing in time: extending a trajectory cannot result in a reduced cost. Therefore, by solving eq. 2.8 iteratively in time, with initial condition $U(\cdot,T) = 0$ at $T = 0$, a sequence of functions $U(x,y,T)$ is obtained which at every point converges monotonically upward to $z(x,y)$ as $T \rightarrow \infty$. Because this convergence is pointwise monotonic, it is clearly stable.

For the actual implementation, an iterative procedure is used that is justified by its exact relation to a discretised control problem [7]. The continuous image plane

is replaced by an image discretized into pixels, and the trajectory described by eq. 2.3 is approximated by a Markov process. This is described in detail in the next section. It can be shown that this gives a discrete approximation U^h to U, which converges to the continuous U as the spatial grid size h approaches zero [11]. However, a naive discretization of eq. 2.8 does *not* necessarily give a stably convergent algorithm.

3 Algorithm Description

A more detailed description of the algorithm and its derivation is now presented. We consider a control problem defined on the discrete grid of pixels and chosen to approximate the continuous calculus of variations problem described above. h is the pixel spacing. For the discrete case, a 'particle' trajectory is a sequence of discrete jumps between grid sites—a poor approximation to a continuous trajectory. In order to better approximate a continuous trajectory on a discrete grid, an element of randomness is introduced.

The control problem is as follows: a 'particle', with initial image plane location $\phi_0 \equiv (i_0, j_0)h$, jumps between neighboring pixel sites in response to control parameters $\mathbf{C} \equiv (u(k), v(k))$, where k indicates the time step. (A 4–neighborhood is assumed.) The jumps are probabilistic, but it is required that on average

$$\langle \phi(k+1) - \phi(k) \rangle = \Delta t \mathbf{C}(k), \tag{3.10}$$

in analogy with eq. 2.3. Here Δt is the time increment from time step k to step $k+1$, and $< >$ denotes the noise average. Let $\eta(k)$ be the random vector representing the jump at time k: $\eta \equiv \phi(k+1) - \phi(k)$. The jump probabilities are assigned as follows. When $u = v = 0$, $\mathbf{P}(\eta = \mathbf{0}) = 1$, with all other probabilities zero. In this case, Δt is arbitrarily chosen to be 1. Otherwise,

$$\begin{aligned} \mathbf{P}(\eta = (\mathrm{sgn}(u),\ 0)h) &= \Delta t|u|/h &&= |u|/(|u|+|v|) \\ \mathbf{P}(\eta = (0,\ \mathrm{sgn}(v))h) &= \Delta t|v|/h &&= |v|/(|u|+|v|), \end{aligned} \tag{3.11}$$

again with all other probabilities zero. Eq. 3.11 implies that the particle jumps by one lattice site at each iteration (for a nonzero control), and therefore moves on the lattice with maximum speed, causing the algorithm to converge quickly. This has been achieved by taking Δt to depend explicitly on the controls as $\Delta t = h/(|u|+|v|)$. It is clear that eq. 3.11 implies eq. 3.10.

The analog to the calculus of variations problem is as follows: choose the control parameters to minimize the *expected* cost for the discrete trajectories:

$$U^h(\phi_0, K) = \inf\{ \left\langle \tfrac{1}{2} \textstyle\sum_{k=0}^{K} \Delta t\,(u(k), v(k))\,(u(k)^2 + v(k)^2 + V(\phi(k))) \right\rangle\}, \tag{3.12}$$

where the infimization is over all nonanticapative control sequences $\{(u(k), v(k)), k = 0, \ldots K\}$ (i.e., controls which do not depend on the future history of the particle) [7]. It can be shown the value function U^h for this discrete control problem converges to the continuous value function as the grid spacing is taken to zero [11].

A dynamical programming equation can be derived for this control problem as in the previous section:

$$\begin{aligned} U^h(\phi_0, K) = \inf_{(u(0),v(0))}\{ & \\ & \tfrac{1}{2}\Delta t(u(0), v(0))\,(u(0)^2 + v(0)^2 + V(\phi_0)) \\ & + \langle U^h(\phi_0 + \eta, K-1) \rangle\}. \end{aligned} \tag{3.13}$$

The expectation in this equation is easily calculated from eq. 3.11; for nonzero controls it is

$$\frac{1}{h}\left(|u|U^h(\phi_0 + (\mathrm{sgn}(u), 0)) + |v|U^h(\phi_0 + (0, \mathrm{sgn}(v)))\right).$$

Performing the minimization in eq. 3.13 is slightly complicated since the cases with $\mathbf{C}$ in different quadrants must be treated separately. Eventually, the following algorithm is obtained. Define

$$\begin{aligned} U^h_{N1} &= \mathrm{Min}(U^h(\phi \pm (1,0)h), \\ U^h_{N2} &= \mathrm{Min}(U^h(\phi \pm (0,1)h), \end{aligned}$$

and let $D_N \equiv U^h_{N2} - U^h_{N1}$. Further, let

$$T \equiv \begin{cases} \tfrac{1}{2}\left(2V - D_N^2\right)^{1/2} & \text{if } V(\phi) > D_N^2 \\ V^{1/2}(\phi) + \mathrm{Min}_i(U^h_{Ni}) & \text{otherwise.} \end{cases} \tag{3.14}$$

The lower case corresponds to the minimum in eq. 3.13 being realized on one of the axes in the u–v plane (with the origin excluded); the upper corresponds to an off–axis minimum. The final update equation is

$$U^h(\phi, K+1) = \mathrm{Min}(T, V(\phi)/2 + U^h(\phi, K)), \tag{3.15}$$

where the second term accounts for the case of zero controls. As in the previous section, the initial value for $U^h(\phi, \cdot)$ should be taken as 0. Since the expected cost for an optimal trajectory cannot decrease with time, an iterative solution $U^h(\phi, K)$ to the above equation increases monotonically at every point. This algorithm is more efficient than the one previously reported in [12] because the time increment Δt is adjusted optimally as a function of the controls.

To avoid indeterminacy, it is necessary to impose the boundary condition that no trajectory exits the image, as is easily done [12]; the significance of this is discussed below. Then, assuming that there are singular points in the image where $V = 0$, as $k \to \infty$ all optimal trajectories must converge to the singular points. Thus $U^h(\phi, K)$ converges monotonically as $k \to \infty$ to a solution $z = U^h(\phi)$; in fact, convergence occurs in a finite number of iterations [11]. This solution satisfies a discretized version of the shape–from–shading equation [12], and is always nonnegative, since the summand in eq. 3.12 is. Also, $z = 0$ at a singular point: a trajectory beginning at a singular point achieves minimal cost by remaining there, since $V = 0$ at the point. Thus, z attains a local minimum at a singular point. Also, as in eq. 2.7, the expected optimal trajectories are approximate curves of steepest descent [12].

The algorithm described in this section is appropriate for *unimodal* images—images containing just one singular point where the height has either a local minimum or maximum. For these images, the iterative solution of eq. 3.15 will correctly reconstruct the original surface at all points where this surface is theoretically determined [13]—that is, at all points connected by a steepest descent curve on the original surface to the singular point. Such points "learn" their height from the singular point. In contrast, at other image points the surface reconstruction can be ambiguous [13]. These ambiguous points lie on steepest descent curves that exit the image rather than terminating at the singular point. Imposing the boundary condition as above that no trajectory exits the image only affects the surface reconstruction at these ambiguous points. Our algorithm does not necessarily reproduce the original surface at ambiguous points. A modified algorithm appropriate for the multimodal case (many singular points) is described in the next section.

4 Modifications of the Algorithm

The algorithm of eq. 3.15 is of the Jacobi type, with the surface updated everywhere in parallel at each iteration. It can be shown that the algorithm also converges if implemented via Gauss–Seidel, with updated surface estimates used as soon as they are available [11]. Our experiments show that this produces a significant speedup.

A more important modification introduces a *terminal cost* term into the cost function. This gives an algorithm capable of dealing with multimodal images. Including this term, the minimal cost is (compare eq. 3.12):

$$U^h(\phi_0, K) = \inf_{(u,v)} \Big\langle g(\phi(K)) + \tfrac{1}{2} \times \sum_{k=0}^{K} \Delta t \, (u(k), v(k)) \left(u(k)^2 + v(k)^2 + V(\phi(k))\right) \Big\rangle . \tag{4.16}$$

The terminal cost, $g(\phi(K))$, introduces a penalty term for a trajectory stopping at the position $\phi(K)$. It causes an optimal trajectory to not remain in regions of high terminal cost, and converge instead to points of low terminal cost. This can dramatically improve the convergence speed of the algorithm. Also, the terminal cost can be used to distinguish between singular points of different type (concave, convex, saddle). In the final surface solution, only a concave–type singular point should be the terminus for optimal trajectories, since these are descending curves. By placing a high terminal cost at other singular points, trajectories can be prohibited from terminating at these points. Then the surface solution will only be "learned" from the concave singular points. Also, if the heights of the concave singular points are known, e.g., using stereo, then this can be specified in the algorithm by setting the terminal costs at these points equal to their heights. Since the singular points are distinctive, it is likely that their heights, and the local nature of the surface, can be determined easily from stereo.

The dynamical programming equation corresponding to the cost in eq. 4.16 is exactly the same as eq. 3.15, as is easily seen. The algorithm differs only in the initial condition for U^h: clearly, U^h should be set initially to $g(\phi)$, not 0 as before. Thus, from the optimal control viewpoint, the choice of initial values for U^h in the algorithm has a concrete and intuitive interpretation.

5 Proof of Equivalence

In this section we will assume the situation of *vertical light*, as described in Section 2. For this case (and under suitable assumptions) the height function has a representation in terms of an associated calculus of variations problem. For the general case of *oblique light* there is a representation in terms of a differential game [11].

The data available for the determination of the function $z(\cdot)$ is encoded in the intensity function $I(x, y)$ determined by eq. 2.1. I is well defined at all points (x, y) where $z(\cdot)$ is differentiable. We will always assume that the function $I(\cdot)$ is defined on a bounded open set of the form $G = \cap_{i=1}^{N} G_i$, $N < \infty$, where each G_i has a C^1 boundary ∂G_i. Let $\hat{n}_i(x, y)$ denote the inward normal to G_i at $(x, y) \in \partial G_i$. First consider the following situation.

Assumption 5.1 *1. $z(\cdot)$ is C^1 on $\overline{G}$.*

2. There is exactly one point $(\tilde{x}, \tilde{y})$ such that $\nabla z(\tilde{x}, \tilde{y}) = 0$.

3. $(\tilde{x}, \tilde{y})$ is a local minimum.

4. $\nabla z(x, y) \cdot \hat{n}_i(x, y) < 0$ whenever $(x, y) \in \partial G \cap \partial G_i$.

(4) implies that the steepest descent direction is always inward on the boundary. We next define a calculus of variations problem. Fix $(x, y) \in \overline{G}$, and set

$$U(x, y) = \inf \int_0^\tau L(\phi(s), \dot{\phi}(s)) ds. \tag{5.17}$$

Here $\tau = \inf\{t : \phi(t) = (\tilde{x}, \tilde{y})\}$, and the infimum is over all piecewise continuously differentiable paths $\phi : [0, \infty) \to \overline{G}$ that satisfy $\phi(0) = (x, y)$. The variational integrand $L(\cdot)$ is given by

$$\begin{aligned} L((x, y), (u, v)) &= \frac{1}{2}\left(u^2 + v^2\right) + \frac{1}{2}\left(\frac{1}{I(x, y)^2} - 1\right) \\ &= \frac{1}{2}\left(u^2 + v^2\right) + \frac{1}{2}|\nabla z(x, y)|^2. \end{aligned}$$

We follow the usual convention of defining $\inf \emptyset = +\infty$. Thus if $\phi(t) \neq (\tilde{x}, \tilde{y})$ for all t, then $\tau = +\infty$.

Theorem 5.2 *Under the conditions of Assumption 5.1 we have*

$$z(x, y) - z(\tilde{x}, \tilde{y}) = U(x, y).$$

Proof. Let $\phi(\cdot)$ be any piecewise continuously differentiable path that starts as (x, y). For all $\epsilon \geq 0$ define

$$\tau^\epsilon = \inf\{t : |\phi(t) - (\tilde{x}, \tilde{y})| \leq \epsilon\}.$$

Fix $\delta > 0$ and choose $\epsilon > 0$ such that

$$z(x, y) \leq z(\tilde{x}, \tilde{y}) + \delta$$

for $|(x, y) - (\tilde{x}, \tilde{y})| \leq \epsilon$.

To prove $z(x, y) - z(\tilde{x}, \tilde{y}) \leq U(x, y)$, we consider two cases. First assume $\tau^\epsilon = +\infty$. By Assumption 5.1 there exists $c > 0$ such that

$$L((x, y), (u, v)) \geq c$$

for all (x,y) satisfying $|(x,y)-(\tilde{x},\tilde{y})| \geq \epsilon$ and all $(u,v) \in \mathbf{R}^2$. Thus, in such a case

$$\int_0^\tau L(\phi(s),\dot{\phi}(s))ds = \int_0^{\tau^\epsilon} L(\phi(s),\dot{\phi}(s))ds = +\infty.$$

Next assume $\tau^\epsilon < \infty$. By the chain rule,

$$\frac{d}{dt}[-z(\phi(t))] = -\nabla z(\phi(t))\cdot\dot{\phi}(t) \leq \frac{1}{2}|\dot{\phi}(t)|^2 + \frac{1}{2}|\nabla z(\phi(t))|^2$$

almost surely in t. Therefore

$$\begin{aligned} z(x,y) - z(\tilde{x},\tilde{y}) &\leq z(x,y) - z(\phi(\tau^\epsilon)) + \delta \\ &= -[z(\phi(\tau^\epsilon)) - z(\phi(0))] + \delta \\ &= \int_0^{\tau^\epsilon} -\nabla z(\phi(t))\cdot\dot{\phi}(t)dt + \delta \\ &\leq \int_0^{\tau^\epsilon} L(\phi(t),\dot{\phi}(t))dt + \delta \\ &\leq \int_0^{\tau} L(\phi(t),\dot{\phi}(t))dt + \delta. \end{aligned}$$

Sending $\delta \to 0$ we obtain $z(x,y) - z(\tilde{x},\tilde{y}) \leq U(x,y)$.

To prove $z(x,y) - z(\tilde{x},\tilde{y}) \geq U(x,y)$, let $\phi(\cdot)$ be a solution (note that there may not be uniqueness since z is only assumed C^1) to the equation

$$\dot{\phi}(t) = -\nabla z(\phi(t)),\ \phi(0) = (x,y).$$

By Assumption 5.1 $\phi(\cdot)$ never touches ∂G for $t > 0$, and therefore the solution is well defined for all $t > 0$. Let $\tau = \inf\{t : \phi(t) = (\tilde{x},\tilde{y})\}$ and let $a \wedge b$ denote the smaller of a and b. For any $t < \infty$, we have

$$\begin{aligned} z(x,y) - z(\tilde{x},\tilde{y}) &\geq z(\phi(0)) - z(\phi(t\wedge\tau)) \\ &= -[z(\phi(t\wedge\tau)) - z(\phi(0))] \\ &= \int_0^{t\wedge\tau} -\nabla z(\phi(s))\cdot\dot{\phi}(s)ds \\ &= \int_0^{t\wedge\tau} |\nabla z(\phi(s))|^2 ds \\ &= \int_0^{t\wedge\tau} L(\phi(s),\dot{\phi}(s))ds. \end{aligned}$$

Sending $t \to \infty$ we conclude that

$$z(x,y) - z(\tilde{x},\tilde{y}) \geq \int_0^\tau L(\phi(s),\dot{\phi}(s))ds \geq U(x,y).$$

■

The solution to the calculus of variations problem uniquely identifies the height function up to an overall translation in z. This ambiguity can be removed by specifying $z(\tilde{x},\tilde{y})$.

We next consider a more general situation involving more than one stationary point. Let M be the set of local minima of $z(\cdot)$.

Assumption 5.3 *1. $z(\cdot)$ is C^1 on $\overline{G}$.*
2. The value of $z(\cdot)$ is known on M.
3. $\nabla z(x,y)\cdot\hat{n}_i(x,y) < 0$ whenever $(x,y) \in \partial G \cap \partial G_i$.

Define the terminal cost function

$$g(x,y) = \begin{cases} z(x,y) & \text{if } (x,y) \in M, \\ +\infty & \text{otherwise.} \end{cases}$$

Consider the calculus of variations problem

$$U(x,y) = \inf\left[\int_0^\tau L(\phi(s),\dot{\phi}(s))ds + g(\phi(\tau))\right]. \quad (5.18)$$

Here the infimum is over all $\tau < \infty$ and absolutely continuous paths $\phi : [0,\tau] \to \overline{G}$ that satisfy $\phi(0) = (x,y)$. Unlike the case of a single stationary point, it is necessary that a terminal cost be included in order to guarantee that trajectories do not get "stuck" at stationary points that are not local minima.

We have the following result for this case.

Theorem 5.4 *Under the conditions of Assumption 5.3 we have*

$$z(x,y) = U(x,y).$$

Remarks on the proof. The proof is very similar to that of Theorem 5.2 and will only be sketched. Consider any path $\phi(\cdot)$ which starts at (x,y) and for which the cost

$$\int_0^\tau L(\phi(s),\dot{\phi}(s))ds + g(\phi(\tau)) \quad (5.19)$$

is finite. Boundedness of the cost implies $\phi(\tau) \in M$. Suppose $\phi(\tau) = (\tilde{x},\tilde{y})$. The proof of Theorem 5.2 then shows that $z(x,y) - z(\tilde{x},\tilde{y}) \leq \int_0^\tau L(\phi(s),\dot{\phi}(s))ds$. Together with the definition of $g(\cdot)$ this implies $z(x,y) \leq U(x,y)$.

Next consider the reverse inequality. As in the proof of Theorem 5.2, we would like to construct a particular path $\phi(\cdot)$ that starts at (x,y) so that the cost (5.19) is arbitrarily close to $z(x,y)$. We first note that by a perturbation argument [13,10,15] we can assume that there are at most finitely many points such that $\nabla z(x,y) = 0$. It can be shown that there exists a dense subset D of $\overline{G}$ with the property that whenever the path $\phi(\cdot)$ satisfies

$$\dot{\phi}(t) = -\nabla z(\phi(t)), \phi(0) \in D,$$

then $\phi(t)$ converges to a local minimum $(\tilde{x},\tilde{y})$ of $z(\cdot)$ as $t \to \infty$. Using the argument of Theorem 5.2 and the fact that $z(\phi(t))$ is nondecreasing we conclude $z(x,y) \geq U(x,y)$ for $(x,y) \in D$. By continuity of both $z(\cdot)$ and $U(\cdot)$ (which is easy to prove) we have $z(x,y) \geq U(x,y)$ for $(x,y) \in \overline{G}$. ■

Previous uniqueness proofs [1,17,14] assumed that $z(x,y)$ was at least C^2; here z is only assumed C^1. A fortiori, no conditions are placed on the second derivatives of the intensity; in particular, the singular points are not required to be "good" or "nondegenerate" [17,14].

6 Illumination from a General Direction

For a Lambertian surface, the image irradiance equation for the intensity is:

$$I(x,y) = \hat{L}\cdot\frac{(-z_x,-z_y,1)}{(1+z_x^2+z_y^2)^{1/2}},$$

where $\hat{L}$ is a unit vector giving the light source direction, and z_x, z_y are partial derivatives of the height. For simplicity and w.l.o.g., we take the x–component of $\hat{L}$

to be zero. After some algebra, this equation may be rewritten as:

$$I^2 z_x^2 + J z_y^2 + 2L_x L_y z_y + (I^2 - L_z^2) = 0,$$

with $J(x,y) \equiv I^2(x,y) - L_y^2$.

Define a new variable $\xi \equiv (x,y,z) \cdot \hat{L}$ measuring the "height" along the light direction rather than the viewer direction $\hat{z}$. This is done so that the local cost at singular points will be zero, like V previously, causing optimal trajectories to terminate at these points. Then

$$\xi_x = L_x z_x, \qquad \xi_y = L_y + L_z z_y.$$

Substituting in the previous equation yields

$$I^2 \xi_x^2 + J \xi_y^2 + 2(1 - I^2) L_y \xi_y - (1 - I^2) = 0.$$

J, the coefficient of ξ_y^2, is positive in an image region B that includes the singular points. When $I^2 = L_y^2$, the angle between the surface normal and $\hat{L}$ is large enough that it may correspond to a point on the occluding boundary.

In the image region B, we consider a control problem analogous to that of Section 2: a 'particle' initially located at $(x,\ y)$ is controlled using the parameters u, v:

$$\dot{x} = I^2(x,y)u, \qquad \dot{y} = J(x,y)v - (1 - I^2(x,y))L_y.$$

u, v are chosen to infimize a cost function for the particle's trajectory

$$U(x,y,T) = \inf_{(u,v)}$$

$$\tfrac{1}{2} \int_0^T ds(I^2(x,y)u^2 + J(x,y)v^2 + (1 - I^2(x,y))). \tag{6.20}$$

As before the integrand is nonnegative throughout the region B, and the local cost $1 - I^2$ vanishes at singular points. Eq. 2.4 for the vertical–light case $L_y = 0$ can be recovered by dividing the above equation by I^2.

This control problem is essentially equivalent to the one previously considered, and results similar to those of the previous sections are easily obtainable. In particular, by a Schwarz inequality argument as in Section 5,

$$\begin{aligned} \frac{d}{dt}\left[-\xi(\phi(t))\right] &= -\nabla\xi(\phi(t)) \cdot \dot{\phi}(t) \\ &= -\xi_x I^2 u - \xi_y \left(Jv - (1 - I^2)L_y\right) \\ &\leq \frac{1}{2}(I^2 u^2 + Jv^2 + \\ &\quad I^2 \xi_x^2 + J\xi_y^2 + 2\xi_y(1 - I^2)L_y) \\ &= \frac{1}{2}\left(I^2 u^2 + Jv^2 + 1 - I^2\right), \end{aligned}$$

which is just the integrand of eq. 6.20. This gives the necessary generalization for the rigorous proof of equivalence. Similarly, an algorithm can be defined in the same way as before, and will recover the correct solution near concave (or convex) singular points.

In the image region where $I^2 - L_y^2 < 0$, the optimal control representation of the problem no longer suffices. Instead, there is a representation in terms of a *differential game* (see e.g. [2]). However, it is a particularly simple one, in which the opposing controllers effectively direct the 'particle' motion in orthogonal directions, and where the cost also splits into a sum of terms depending on the different control parameters. Thus, the Isaacs condition and the existence of a "value" follow.

The 'particle' dynamics for the differential game is:

$$\dot{x} = I^2 u, \qquad \dot{y} = J\left(\theta(J)v_1 + \theta(-J)v_2\right) - (1 - I^2)L_y,$$

where

$$\theta(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases}$$

The player associated with u and v_1 seeks to minimise the value function of the game, while the v_2 player seeks to maximise it. The value that opposing players attempt to control is

$$U(\phi,t) \equiv \tfrac{1}{2} \times$$

$$\int_0^T ds(I^2 u^2 + J(\theta(J)v_1^2 + \theta(-J)v_2^2) + (1 - I^2)).$$

A precise description of the differential game is somewhat technical (see e.g. [2]). Here we simply note that the properly defined value gives the height function (under suitable conditions), and that an algorithm on a discrete grid for approximating this value function can be derived that is similar to the vertical–light algorithm.

7 Experiments

Figure 1 displays a 32 by 32 surface parabolic surface which is assumed to be imaged from above. The image has one singular point. Assuming vertical light, the image intensity was first computed using the discretization of the derivative implicit in eq. 3.15 [12]. With this choice, the original surface is a fixed point of the algorithm and should be reconstructed exactly. Using Jacobi updates, the algorithm converged to the correct solution to within, on average, one part in 10^7 after 63 iterations. In general, the convergence time is expected to be on the order of the maximum length of an optimal trajectory. Since from eq. 3.11 an optimal trajectory jumps one lattice site per iteration, when the number of iterations becomes greater than the maximum trajectory length, then all image points are able to "learn" their heights from the singular points. For the given surface, the maximal trajectory length is on the order of 32, since trajectories starting at the image corners must zigzag to the singular point at the center of the image.

Convergence using Gauss–Seidel updating was faster: it was obtained after just 17 iterations. The states were updated in a spiral pattern outward from the singular point, since this reflects the approximate information flow. The points near the singular point learn their height early, and this information can then be used in determining the heights of more distant points.

For an image obtained by analytically differentiating the displayed surface, convergence was obtained in the same number of steps. The average and maximal errors were .8 and 1.6 (the latter obtained at the image boundary), compared with a range for the surface height of 25.

The algorithm has also been applied to a noisy image of this surface; the result is a noisy approximation of the surface. The convergence time is longer since the steepest descent curves lengthen due to wiggling. The surface was also reconstructed assuming oblique lighting.

For comparison, Figs. 2, 3 display the result of applying our implementation of Horn's algorithm [4] to a similar surface. The intensity is computed differently than before, using the discrete forward derivatives appropriate for this algorithm. Even after 3072 iterations, the algorithm has not converged to the correct solution. We have also implemented the variational algorithms of [8] and [19], and applied them to this surface with similar results. As also noted by [9], standard variational algorithms often give a wrong, saddle–shaped surface for such simple images containing one singular point.

Figure 4 shows a more complicated 128 by 128 surface. As for Figure 1, the intensity was first computed assuming the discretization of eq. 3.15. The algorithm this time incorporated a terminal cost—an initial value for U—which was large everywhere but at the concave singular points. At these points, U was initialized to the known height values. For vertical light, the algorithm converged to a perfect reconstruction of the original surface in 100 iterations. As expected, the convergence time is on the order of the longest optimal trajectory. When the intensity was derived analytically, the algorithm again converged in 100 iterations, with an average error of 1.6 compared to a surface range of 51 (Figure 5). Because the surface does not obey the boundary condition that it is decreasing in from the boundary (Section 3), the reconstruction is incorrect in places at the boundary, though it is good in the interior. This is clear in Figure 6, which displays the difference between the reconstruction and the original surface. This surface was also reconstructed assuming oblique light at an angle of 17.5° to the vertical. For an intensity derived as for eq. 3.15, convergence to within one part in 10^{-7} was obtained within 120 iterations. Reconstruction for the analytically–derived intensity function was also obtained in about 120 iterations, with an average error of 2.2. As previously, the reconstruction was good in the interior but incorrect along one boundary (Figure 7).

Figure 8 shows the result for vertical light of applying the algorithm without the terminal cost. The algorithm reconstructs a surface that is locally concave at all singular points; it is correct in the neighborhood of those singular points where the surface is in fact locally concave. Note the sharp orientation discontinuities at the boundaries between the regions associated with different singular points.

Finally, our algorithm has been applied to the real 200×200 image shown in Figure 9, which was provided to us by Yvan Leclerc of SRI. The light is from above at $(0, .488, .873)$. For the reconstruction, just one singular point was used, located on the tip of the nose, although the image actually contains several. This has the effect of planing down the surface bumps associated with the other singular points. Figure 10 shows the reconstruction obtained using Gauss–Seidel after 80 iterations, illuminated from the same direction as the original. Figure 11 shows the reconstruction illuminated from below. Convergence has nearly been achieved, apart from small patches near the occluding boundary. This reconstruction took about 2 minutes of CPU time on a DEC 5000 work station. Standard variational algorithms typically require thousands of iterations [4]. Convergence was complete after about 200 iterations; the result, shown in Figure 12, differs little from Figure 11. Finally, Figure 13 shows the surface reconstruction.

For comparison, Figures 14, 15 display the reconstruction obtained by the authors of [8] using a more standard variational method [8], developed for the purpose of including stereo information. Stereo information was used as an initial condition for this reconstruction.

8 Acknowledgments

We thank Yvan Leclerc for providing the image and reconstruction displayed in Figures 9, 14, 15, and for enjoyable conversations.

References

1. A. R. Bruss, "The Eikonal Equation: Some Results Applicable to Computer Vision," *Journal of Math. Phys.* 23(5): 890-896, May 1982.
2. R. J. Elliott, *Viscosity Solutions and Optimal Control*, Longman Scientific and Technical, New York, NY: 1987.
3. B.K.P. Horn, "Obtaining Shape from Shading Information," in *The Psychology of Computer Vision*, P. H. Winston (ed.), McGraw Hill: New York, 1975, pp. 115-155.
4. B.K.P. Horn, "Height and Gradient From Shading," *International Journal of Computer Vision* Vol. 5 No. 1, pp. 37–75, 1990.
5. B. K. P. Horn and M. J. Brooks, "The Variational Approach to Shape from Shading," *Computer Vision, Graphics, and Image Processing*, Vol. 33, pp. 174–208,1986.
6. K. Ikeuchi and B. K. P. Horn, "Numerical Shape from Shading and Occluding Boundaries," *Artificial Intelligence*, Vol. 17, Nos. 1–3, pp. 141–184, August 1981.
7. H. J. Kushner, "Numerical methods for stochastic control problems in continuous time,", *SIAM J. Control and Optimization*, Vol. 28, pp. 999–1048, 1990.
8. Y. G. Leclerc and A. F. Bobick, "The Direct Computation of Height from Shading," *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Lahaina, Maui, Hawaii, 1991, pp. 552-558.
9. Y. G. Leclerc and A. F. Bobick, *personal communication.*
10. J. Milnor, *Morse Theory*, Annals of Mathematics Studies 51. Princeton University: New Jersey, 1970.
11. J. Oliensis and Paul Dupuis, "Direct method for reconstructing shape from shading," in preparation.

12. J. Oliensis and Paul Dupuis, "Direct method for reconstructing shape from shading," *Proc. SPIE Conf. 1570 on Geometric Methods in Computer Vision*, San Diego, California, July 1991, pp. 116-128.

13. J. Oliensis, "Shape from Shading as a Partially Well-Constrained Problem," *Computer Vision, Graphics, and Image Processing: Image Understanding*, Vol. 54, No. 2, pp. 163-183, 1991.

14. J. Oliensis, "Uniqueness in Shape From Shading," *The International Journal of Computer Vision*, Vol 6, No. 2, pp. 75-104, 1991.

15. J. Palis and W. de Melo, *Geometric Theory of Dynamical Systems*. Springer-Verlag: NY, 1982.

16. B. V. H. Saxberg, "An Application of Dynamical Systems Theory to Shape From Shading," in *Proc. DARPA Image Understanding Workshop*, Palo Alto, CA, May 1989, pp. 1089-1104.

17. B. V. H. Saxberg, "A Modern Differential Geometric Approach to Shape from Shading," MIT Artificial Intelligence Laboratory, TR 1117, 1989.

18. E. Rouy, A. Tourin, "A Viscosity Solutions Approach To Shape-From-Shading," June 1991, to appear in *SIAM J. on Numerical Analysis*.

19. R. Szeliski, "Fast Shape from Shading," *Computer Vision, Graphics, and Image Processing: Image Understanding*, Vol. 53, pp. 129-153, 1991.

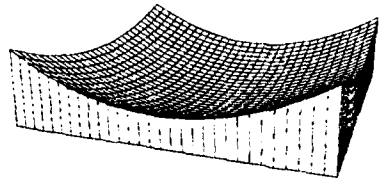


Fig. 1 Parabolic surface.

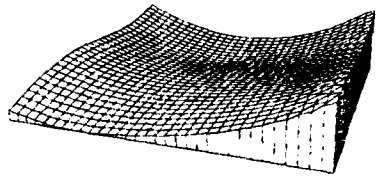


Fig. 2 Horn's algorithm: 128 iterations.

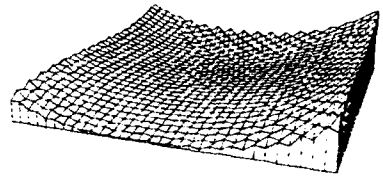


Fig. 3 Horn's algorithm: 3072 iterations.

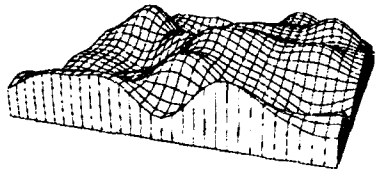


Fig. 4 Complex surface.

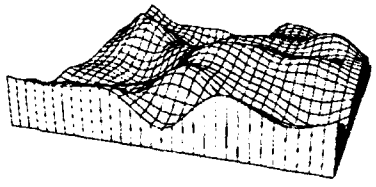


Fig. 5 Reconstruction.

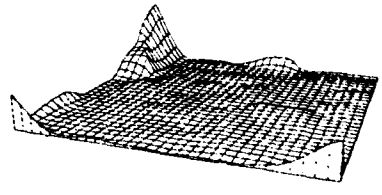


Fig. 6 Difference surface.

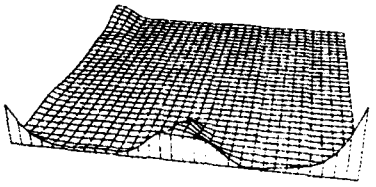


Fig. 7 Difference surface.

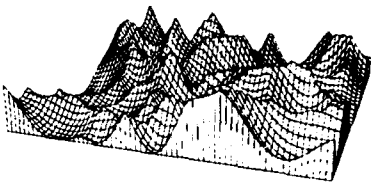


Fig. 8 Result with no terminal cost.

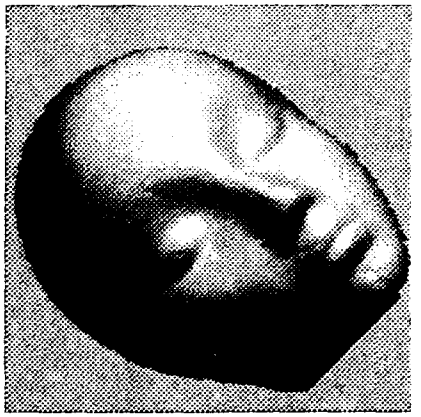


Fig. 9 Mannequin image.



Fig. 10 Reconstruction lighted from above.



Fig. 11 Reconstruction lighted from below.



Fig. 12 Final reconstruction.

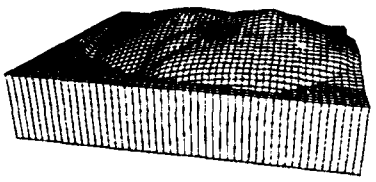


Fig. 13 Surface reconstruction.

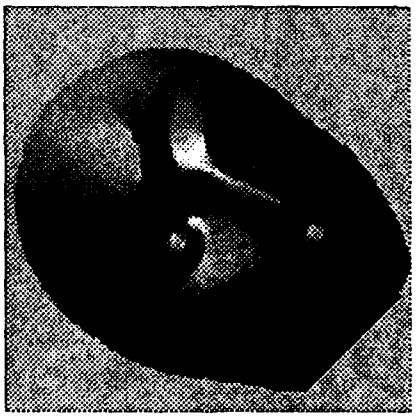


Fig. 14 Reconstruction [8] lighted from above.

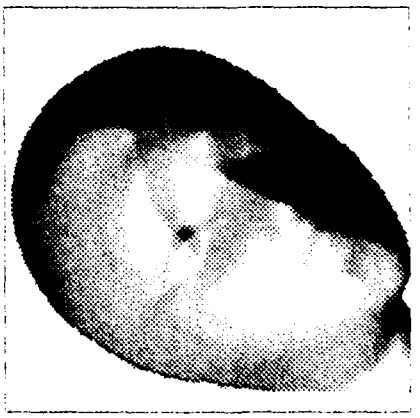


Fig. 15 Reconstruction [8] lighted from below.

Extracting Geometric Shapes From A Set Of Points

Alicia Abella
Department of Computer Science
Columbia University
New York, NY 10027

Abstract

This paper details the use of mathematical techniques from computational geometry and fuzzy set theory, for the geometric classification of a set of points. This investigation was encouraged by a study of topological visual navigation in two-dimensional spaces. The navigational system employed extracts information from the environment in the form of landmarks and represents these landmarks as points in an abstracted map of the original environment. Left with only point-like objects of the environment it is necessary to describe the point-like objects in terms of their interrelationship. This paper describes why point-like objects were chosen to represent the environment, why the point-like objects are described by the geometrical shapes they form, and just how these tasks are performed. The use of the convex hull of a set of points and the use of membership functions from fuzzy set theory are employed to classify the point-like objects into such basic geometric shapes as triangles, squares, rectangles, and other n-sided polygons, such as pentagons, hexagons and octagons. Pattern recognition, spatial analysis, and topological navigation, are areas that can benefit from this study.

1 Introduction

The problem of extracting the geometric shape from a set of points arose from work in the area of topological visual navigation in two-dimensional spaces [5]. Traditional visual navigation has depended primarily on quantitative measures that are captured in the form of metric maps. Direction-giving in such a system relies on the metric map for travel distance and directional information. Direction-giving in the topological visual navigation system employed in this paper, relies on landmarks instead of metric maps. A landmark is [5]

> an object that can be recognized by a navigator along a path ... regardless of the intervening object nodes. Thus a tree in a desert is a landmark, but a tree in a forest is not.

These landmarks are viewed aerially, as from a helicopter, therefore there is no occlusion of landmarks and an abundance of information regarding the environment can be extracted. However, much of this information can be disregarded and considered extraneous for the navigation problem. Much of the aerial view of an environment consists of empty space between objects. This fact can be used to direct navigation according to the arrangement of the objects, rather than by their distance. In this case an object - or interchangeably a landmark - can be considered a point on our map of the environment.

It would seem, off hand, that reducing our environment to point-like objects simplifies our problem, but it in fact creates some new ones. Attributes such as size, color and shape are no longer available. We no longer can describe an object as "the big red block". Instead we have a set of point-like objects with no size, color or shape distinction. Nevertheless, the choice of reducing our problem to point sets is made because any object can be represented as a point and it leads to an intuitively appealing way of characterizing geometric form. It further allows a qualitative and quantitative way of describing shape form. The approach in this paper has been a qualitatively descriptive one, as opposed to a very rigorous quantitative approach.

There are several reasons for choosing a qualitative description of form: it is less prone to error, and more intuitive than a quantitative description. It is less prone to error because the shape form does not rely solely on metrics for its description. It is more intuitive because people in essence do not speak in terms of inches and degrees but rather in terms of shapes and their spatial orientation. As an example, consider the problem of asking a person to single out a point among a set of points. It is more likely that the person will describe the point with regard to those points around it , than to describe its exact coordinate location. The description given might be any of the following: most isolated point, directly north of the bottom-most point, second point from the top, in the middle of the points forming the square, etc.

After simplifying our environment to include only point-like objects, the next step is the actual characterization of these point sets into some geometric shape. It is beneficial to characterize points into geometric shapes because they are the building blocks of more complex shapes, and such a classification is invariant to rotation, scale, and translation. As mentioned earlier, to qualitatively describe points it is necessary to study their relationship with those points around them. The approach in this paper is to describe points as they interrelate geometrically. Other approaches to point set analysis include [6] which describes a methodology for describing the internal shape of a set of points by using a measure of neighborliness of points. The paper [1] developed a generalization of the convex hull to describe the external shape of a set of points. The paper [2] developed techniques to analyze sparse images, which

are represented as sets of points in two-dimensional space. The paper [9] investigates the problem of connecting dots into orthogonal polygons. The following sections go on to describe how to classify a finite set of points into geometric shapes, such as triangles, squares, rectangles, and other n-sided polygons, such as pentagons, hexagons, and octagons, using techniques from computational geometry and fuzzy set theory.

2 The Basic Theoretical Foundation

As explained in the introduction, this paper aims to classify the shape of a set of points using the convex hull of a point set and membership functions from fuzzy set theory. For completeness sake, this section will begin with a brief review of the convex hull and fuzzy set theory. The following sections will contain a more detailed description of the actual membership functions used in ultimately determining the geometrical shape of a set of points.

2.1 The Convex Hull Of A Set Of Points

The first step in characterizing the geometrical shape of a set of points is to determine the convex hull of the point set. The convex hull of a set of points consists of the extreme points of the set, and as such, it provides a suitable description of the shape of the point set, as shown in figure 1. The weakness of the convex hull is in choosing the extreme points as representative of essential points. In other words, the extreme points may not necessarily be the best descriptors of the shape of the point set; some interior points may be better suited to be the descriptors of shape. The convex hull, however, has been well-studied, there are many efficient algorithms for computing it, and it provides the necessary properties for classifying the shape of a point set and it is invariant to rotation, translation, and scale. It is for these properties that the convex hull is employed to tackle the task of classifying the geometrical shape of a set of points.

2.2 A Brief Overview Of Fuzzy Set Theory

The theory of fuzzy sets is used to represent uncertainty, information, and complexity [7]. The theory of classical sets [1], on the other hand, represents certainty. A classical set divides the world into two groups: those that certainly belong to a set and those that certainly do not belong to a set. A fuzzy set, on the other hand, divides the world much more loosely, by introducing vagueness into the grouping process. This means that members of a set belong to that set to a greater or lesser degree than other members of the set. Mathematically, members of the set are assigned a membership grade value that indicates to what degree they belong to the set. This membership grade is usually a real number in the closed interval be-

[1] Also know as crisp sets.

tween 0 and 1. Therefore a member that has a membership grade closer to 1 belongs to the set to a greater degree than a member with a lower membership grade. Because of its properties, fuzzy set theory, can find application in fields that study how we assimilate information, recognize patterns, and simplify complex tasks.

The theory of fuzzy sets applies quite naturally to the task set forth in this paper because we seek "approximate" measurements and not "exact" ones for determining the geometrical shape of our point set. In other words a rectangular-like figure with angles 89°, 88°, 88°, and 93° should be classified as a rectangle, even though the angles aren't all exactly 90°. The following section will describe just how to make such a classification.

3 The Geometric Shape Of A Set Of Points

The first step in extracting the geometric shape from a set of points is to compute the convex hull of the points. Papers [4, 3] each detail an algorithm for computing the convex hull. The number of points of the convex hull then dictate the course of actions to follow. Two points obviously describe a line. Three points can describe a right triangle, isoceles triangle, equilateral triangle, and ordinary triangle. Four points can describe a square, rectangle, rhombus, and a trapezoid. There is no specific classification for n-sided polygons where n is greater than four, but such polygons can be classified as pentagons, hexagons, heptagons, and octagons. As mentioned in section 2.2, membership functions allow for "approximate" measurements, and as such they allow for the classification of "approximate" polygons. The paper[8] provides the various membership functions that will allow the classification of the forementioned geometric shapes. The following subsections will present these membership functions and describe how they are used to classify the "approximate" geometric shape of a set of points.

3.1 Triangle Classification

As mentioned in the introduction to this section, a triangle can have various classifications: right, isoceles, equilateral, and ordinary. To determine which of the various triangle classifications a set of three points can form, the following membership functions are used, see also figure 2:

$$\mu_{right} = 1 - \rho_{right}\ min\{|A - 90°|, |B - 90°|, |C - 90°|\}/180°$$

$$\mu_{isoceles} = 1 - \rho_{isoceles}\ min\{|A - B|, |B - C|, |C - A|\}/180°$$

$$\mu_{equilateral} = 1 - \rho_{equilateral}\ max\{|A - B|,$$

$$|B - C|, |C - A|\}/180°$$

A, B and C are the angles formed between the points of the convex hull, the various μ's are the membership grades and the ρ's are constants.

To determine the constants each membership function is computed by choosing angles A, B , and C to be those angle measurements that are least like any of the triangles. For example, a "triangle" with angles A = 180°, B = 0°, and C = 0° can be assumed to be most unlike a right triangle, therefore

$$\begin{aligned} \mu_{right} &= 0 = 1 - \rho_{right}\ 90/180 \\ &\Longrightarrow \rho_{right} = 2 \end{aligned}$$

Adopting the same approach for the other constants yields $\rho_{isoceles} = 3$ and $\rho_{equilateral} = 1$. To determine if three points form any of the forementioned triangles a threshold value δ must be chosen. The threshold value is chosen according to the degree of exactness required for the point classification. A greater threshold value demands a more exact classification and a lower threshold value demands a looser classification.

After applying the angles to each of the membership functions the one with a membership grade greater than the threshold value is chosen to be the triangle that most closely characterizes the set of points. If more than one membership grade is greater than the threshold than there is a possibility that the triangle should be classified as belonging to both sets. Various functions can be created to test for this property. One such function would calculate the product of both membership grades. If the result yields a value greater than the threshold than the triangle should be classified as belonging to both sets. For example if $\mu_{isoceles} = 0.9$ and $\mu_{right} = 0.9$ their product is 0.81 and the triangle should be classified as a right isocleses triangle. If, however, the result is not greater than the threshold than the maximum of the membership grades is chosen. If none of the membership grades is greater than the threshold value then the set of points is considered to form an ordinary triangle.

3.2 Quadrangle Classification

As mentioned in the introduction to this section, four points can have various classifications: trapezoid, rectangle, rhombus, and square. To determine which, if any, of these various quadrangle classifications a set of four points can form, the following membership functions are used, see also figure 3:

$$\begin{aligned} \mu_{trapezoid} &= 1 - \rho_{trapezoid}\ min\{|A + B - 180°|, \\ &\quad |B + C - 180°|\}/180° \end{aligned}$$

$$\begin{aligned} \mu_{rectangle} &= 1 - \rho_{rectangle}\ \{|A - 90°| + \\ &\quad |B - 90°| + |C - 90°| + \\ &\quad |D - 90°|\}/90° \end{aligned}$$

$$\mu_{rhombus} = 1 - \rho_{rhombus}\ max\{|a-b|, |b-c|, |c-d|, |d-a|\} / a+b+c+d$$

$$\mu_{square} = \mu_{rectangle} \times \mu_{rhombus}$$

A, B, C, and D are the angles formed between the points of the convex hull, the various μ's are the membership grades, the ρ's are constants, and a, b, c , and d are the lengths between the points. The ρ's are calculated the same way they were for the triangle cases. For example, a rectangle with angles A = 180°, B = 180°, C = 0°, and D = 0°, can be assumed to be most unlike a rectangle, therefore:

$$\mu_{rectangle} = 0 = 1 - \rho_{rectangle}\ 360°/90° \Longrightarrow \rho_{rectangle} = 1/4$$

A rhombus requires information on its sides for classification. Therefore to compute $\rho_{rhombus}$, it is assumed that the shape least like a rhombus has length equal to some real value, and length b, c, and d equal to zero, therefore:

$$\mu_{rhombus} = 0 = 1 - \rho_{rhombus} \Longrightarrow \rho_{rhombus} = 1$$

Using the same technique for calculating $\rho_{rectangle}$ yields $\rho_{trapezoid} = 2$. The membership function for a square is the product of the membership grade of a rhombus and rectangle because angles alone and lengths alone are not sufficient to classify the square, information regarding both is needed.

To classify a quadrangle as either a square, rectangle, rhombus, or trapezoid, the same method as the triangle classification is employed - choose a threshold value and find the maximum among the membership grades, if none are greater than the threshold value then the quadrangle is classified as an ordinary four-sided quadrangle.

3.3 N-Sided Polygon Classification

The previous two subsections described the membership functions that allowed for the classification of various triangles and quadrangles. This subsection illustrates the generalized formula for classifying n-sided polygons, and the generalized constant, ρ. The membership function is

$$\mu_{n-sided} = 1 - \rho_{n-sided}\ max\{|A_1 - \frac{(n-2)\times 180°}{n}|, |A_2 - \frac{(n-2)\times 180°}{n}|, \cdots, |A_n - \frac{(n-2)\times 180°}{n}|\}/180°$$

The same approach used to calculate the ρ's for the triangle and quadrangle mem-

bership function is used to calculate the ρ for this function; the shape most unlike the shape we wish to classify is assumed. In the case of this function the result of applying the max portion of the function will always produce

$$\mu_{n-sided} = 0 = 1 - \rho_{n-sided} \times (n-2) \times 180°/n$$

$\rho_{n-sided}$ can be generalized to be

$$\rho_{n-sided} = \begin{cases} \frac{n}{2} & \text{for n = 3} \\ \frac{n}{n-2} & \text{for } n > 3 \end{cases}$$

Using $\rho_{n-sided}$ as just illustrated yields $\rho_{pentagon} = \frac{5}{3}$, $\rho_{hexagon} = \frac{3}{2}$, $\rho_{heptagon} = \frac{7}{5}$, and $\rho_{octagon} = \frac{4}{3}$.

As was the case with the triangle and quadrangle classification, a threshold value is chosen and compared to the membership grade for the particular n points we wish to classify. If the membership function is greater than the threshold value then the n points are classified appropriately, e.g. if the number of points is five then the polygon is classified as being a pentagon. If the membership grade is not greater than the threshold value then the n points are classified as forming a regular n-sided polygon.

A line with n points can also be classified using the membership function approach. The following membership function is employed:

$$\mu_{line} = max\{|A_1 - A_2|, |A_2 - A_3|, \cdots |A_n - A_1|\}/180°$$

Before any of the forementioned membership functions are computed the line membership function should be computed because $\mu_{isoceles}$ and $\mu_{trapezoid}$ will yield a value of one when the points form a line or an "approximate" line. For example, a perfectly straight line of three points consist of angles 0°, 180°, and 0°, when these angles are evaluated using the the isoceles membership function the result is one and the straight line will be classified as an isoceles triangle. An apppropriate threshold value should be selected and once the line membership grade no longer exceeds the threshold value, the other membership functions should be employed.

3.4 Interior Points

The discussion up until this point has centered on the shapes formed by the exterior points of the point set, but mention should also be given to the interior points. Because the focus of this paper is to describe the geometrical shapes of a set of points, the interior points of the convex hull will also be described geometrically. The method is to compute the convex hull of the remaining points and apply the membership functions to determine its geometrical shape. The method is repeated if more than two points remain whose convex hull has not been determined. In this way a nesting of geometrical shapes is produced, as shown in figure 4.

A problem arises when the interior points are not as neatly arranged as shown in figure 4, but rather they look like those of figure 5. The nesting algorithm would classify A in figure 5 as a square within a square and classify B as a seven-sided polygon. It is apparent, however, that such a classification isn't necessarily the most intuitive for the particular arrangment of points. The problem lies in the fact that the shape description has been confined to those that have specific names associated with them, e.g. squares, triangles etc. Description for shapes like those of figure 5 do not structurally exist. What does exist is a manner of describing the general arrangement of their points. For example figure B can be described as consisting of one right angle and a bowl-shaped arrangement of the remaining points.

A possible solution to this problem would be to come up with a grammar for describing those shapes that don't fall under any of the prescribed shapes mentioned so far, but that nevertheless have their points distinctly arranged. Some possible descriptions may include: sharp-angled , curved, S-shaped, L-shaped, bowl-shaped, etc. The list can easily grow to formidable lengths. Obviously some restrictions must be made in governing what grammars to choose. Work in this area is currently in progress.

4 Conclusion

This paper has explored an approach to the problem of extracting the geometric shape from a set of points, in relation to the problem of visual navigation. The natural next step is to devise an approach that can use the ideas put forth in this paper to further describe the point-like objects. For example, a point-like object can be defined as belonging to a right triangle, and can be further described as being that point which forms the right angle. Another step that is currently in the making is that of devising a grammar to describe those shapes that can not be classified by the geometric shapes presented in this paper. It is hoped that the forementioned approaches to point descriptions will lead to a qualitative way of direction-giving in visual navigation. This approach, however, is not limited to problems of visual navigation alone, it is clearly applicable to such areas as pattern recognition and spatial analysis.

References

[1] Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, IT-29(4):551–559, July 1983.

[2] V. Di Gesù. Problems and possible solutions in the analysis of sparse images. *Pattern Recognition Theory and Applications*, F30:277–286, 1987.

[3] R.L. Graham. An efficient algorithm for determining the conex hull of a fi-

nite planar set. *Information Processing Letters*, 1:132–133, 1972.

[4] R.A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2:18–21, 1973.

[5] John R. Kender, Il-Pyung Park, and David Yang. A formalization and implementation of topological visual navigation in two dimensions. *SPIE International Symposia*, 1388:476, November 1990.

[6] David G. Kirkpatrick and John D. Radke. A framework for computational morphology. In G.T. Toussaint, editor, *Computational Geometry*, pages 217–248. Elsevier Science Publishers, 1985.

[7] George J. Klir and Tina A. Folger. *Fuzzy Sets, Uncertainty, and Information.* Prentice Hall, 1988.

[8] E.T. Lee. Proximity measures for the classification of geometric figures. *Journal of Cybernetics*, 2:43–59, 1972.

[9] Joseph O'Rourke. Uniqueness of orthogonal connect-the-dots. In G.T. Toussaint, editor, *Computational Morphology*, pages 97–104. Elsevier Science Publishers, 1988.

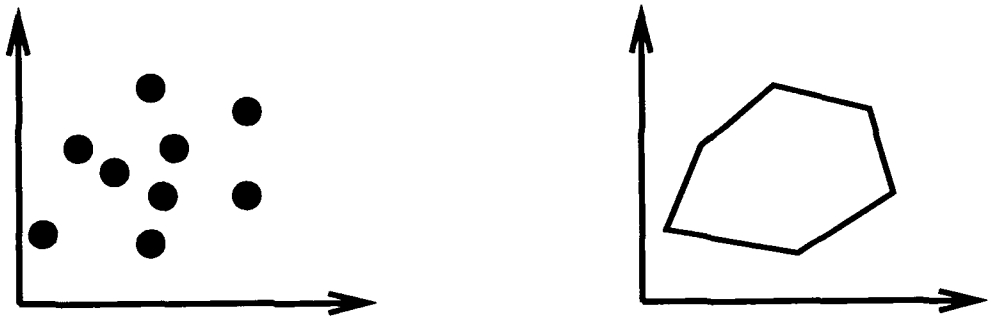


Figure 1: Example Convex Hull

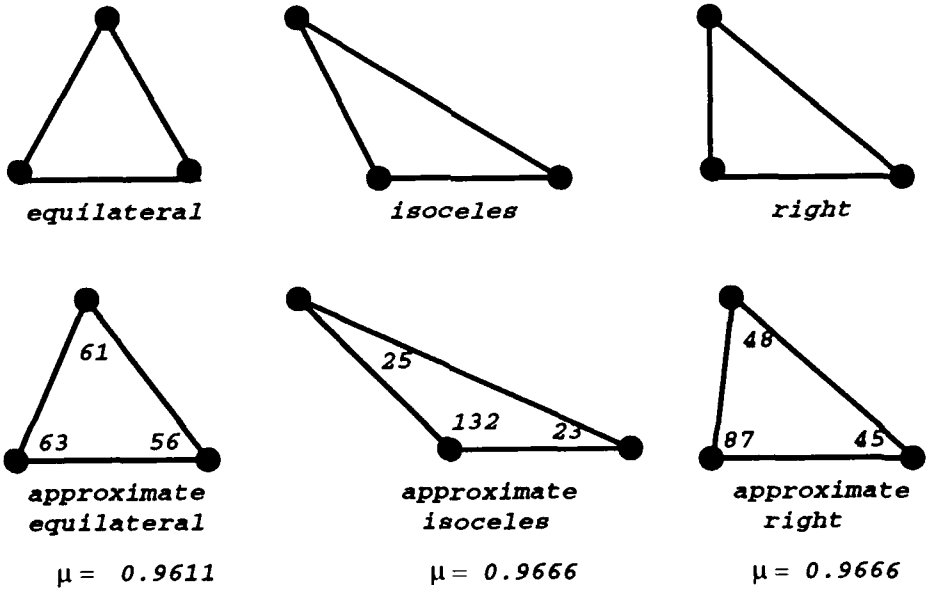


Figure 2: Various Triangle Classifications

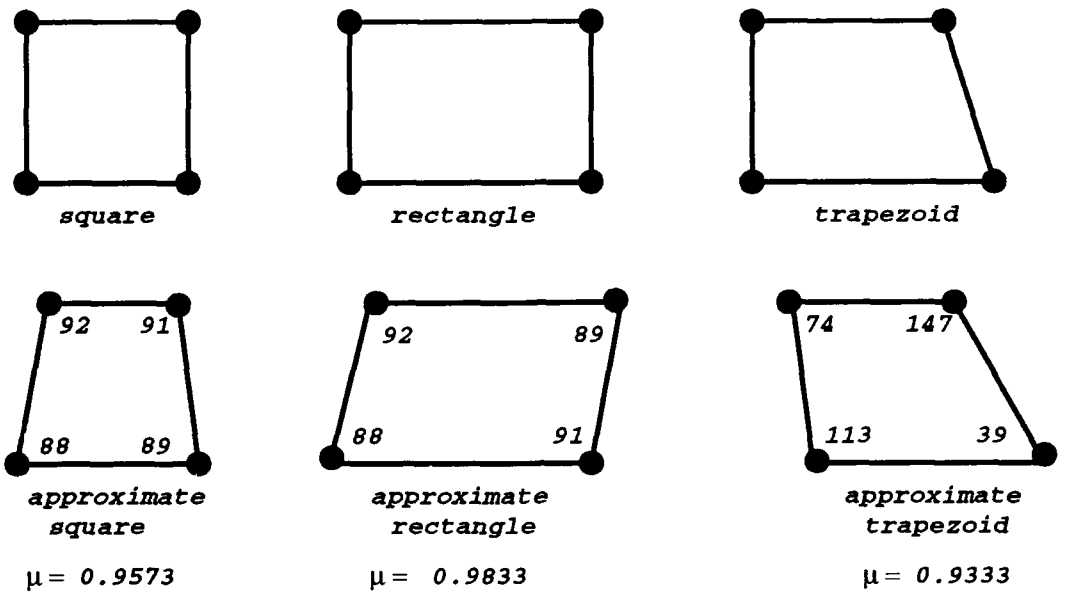


Figure 3: Various Quadrangle Classifications

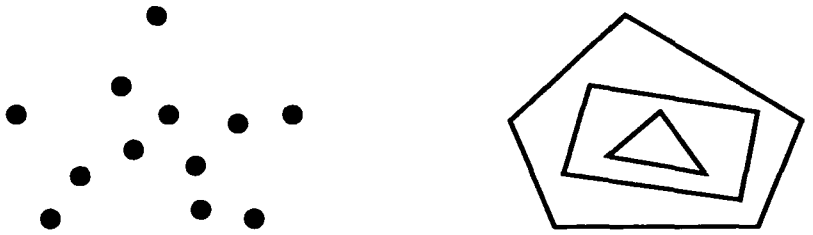


Figure 4: Nested Geometric Shapes

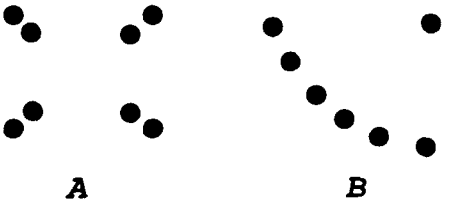


Figure 5: Points With Missing Shape Descriptors

On Seeing Spaghetti:
A Novel Self-Adjusting Seven Parameter Hough Space for Analyzing Flexible Extruded Objects

John R. Kender
Department of Computer Science
Columbia University
New York, NY 10027
kender@cs.columbia.edu

Rick Kjeldsen
Exploratory Computer Vision Group
IBM Thomas J. Watson Research Center
P.O. Box 704 Yorktown Heights, NY 10598
fcmk@watson.ibm.com

Abstract

We present a model for flexible extruded objects, such as wires, tubes, or grommets, and demonstrate a novel, self-adjusting seven-dimensional Hough transform that derives and analyzes their three-space curved axes from position and surface normal information. The method is purely local and is very cheap to compute. The model considers such objects as piecewise toroidal, and decomposes the seven parameters of a torus into three nested subspaces, the structure of which counteract the errors implicit in the analysis of objects of great size and/or small curvature. We believe it is the first example of a parameter space structure designed to cluster ill-conditioned hypotheses together so that they can be easily detected and ignored. This work complements existing shape-from-contour approaches for analyzing tori: it uses no edge information, and it does not require the solution of high-degree non-linear equations by iterative techniques. Most of the results, including the conditions for the existence of more that one solution (phantom "anti-tori"), have been verified using a symbolic mathematical analysis system. We present, in the environment of the IBM ConVEx system, robust results on both synthetic CAD-CAM range data (the hasp of a lock), and actual range data (a knotted piece of coaxial cable), and discuss several system tuning issues.

1. Introduction

We consider the problem of analyzing dense depth images to determine the parameters of flexible extruded objects. Our approach views such objects as being piecewise toroidal, since the torus is the simplest solid geometric object whose generating axis exhibits curvature. This is justified by our observation that many piecewise sections of flexible extruded objects have co-planar spines, either because they are easy to manufacture that way, or because they represent a minimum energy configuration (these reasons are often equivalent). However, a torus is an object with seven free parameters: three of position, two of orientation, one of size, and one of relative "thickness". This pushes the limits of the current state of the art in parsing high-degree parametric object models, particularly since the parameters of position and size are potentially unbounded, especially for objects that are nearly cylindrical.

Following what might be called the principle of least variability, we first recover the thickness parameter, since among all seven it is the one parameter most likely to be constant across the toroidal pieces. The method exploits a novel but efficient algorithmic interpretation of a result in differential geometry called Meusnier's theorem. Then, using the knowledge gained from deriving the thickness, we recover the next most well-behaved parameters, the orientation and size. These are computed simultaneously and in a manner that compensates for the ill-conditioning of orientation estimates when size is large. Lastly, using knowledge of both thickness and size, we recover the position of the local toroidal section, again automatically compensating for the ill-conditioning of large and/or nearly straight objects.

The significance of the work rests on its two principal results: the elegance and speed of the thickness-finding transform (reviewed only briefly here, for more details see [Kender and Kjeldsen, 1990], and the novel way in which the parameter space structure decomposes a difficult seven-dimensional problem so that ill-conditioned hypotheses cluster together for easy detection and removal. The work as a whole is applicable in vision systems wherever depth and surface orientation (however sparse) are obtainable, particularly for those cases where object boundaries are occluded, and where contour-based methods therefore fail.

1.1. The Torus in Brief

We adopt the terminology of DoCarmo [1976], and review the results on tori that we will exploit.

A torus is a solid of revolution formed by a generating circle ("minor circle") of radius r being swept in a circle of revolution ("major circle") of radius a (Note that we have reversed the sense of r & a from DoCarmo). Generally speaking, we will assume that $r<a$, that is, that the "donut" has a "hole". This constraint will be exploited in the decomposition of the parameter spaces.

Surface properties are best represented in the framework of the following parameterization, which identifies the center of the torus (and hence, the center of the major circle) with the

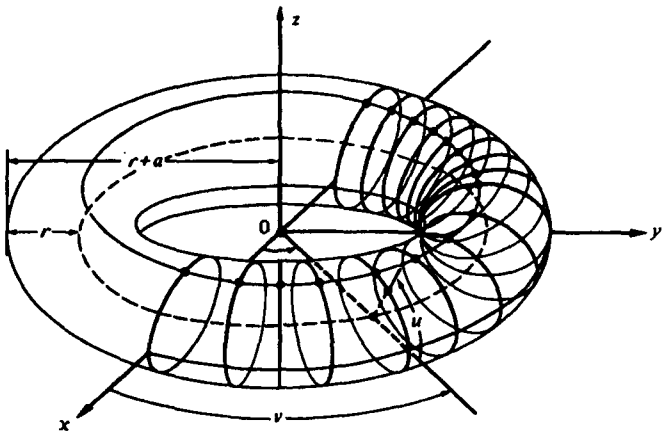


Fig. 1: Torus terminology.

center of a cylindrical coordinate system, making a to be cylindrical radius and v as cylindrical angle; the axis of the torus becomes the remaining coordinate z. We parameterize the surface of the minor circle by the angle u; unlike the angle v, its origin is critical, and is set to equal 0 when it is most distant from the torus center. The locus of points with a given v lie on a "meridian". The locus of points with a given u lie on a "parallel".

As with any other regular surface, each point of the torus has two principal curvatures, with directions of principal curvature that are perpendicular to each other. Every point has one positive curvature, k_1, which is oriented along the meridian. The other curvature of the point, k_2, is more elaborate, although it is always oriented along the parallel. k_2 is 0 at the "top" and "bottom" parallels of the torus, where the surface is locally cylindrical. It attains its extreme positive value at the "outermost" parallel, and its extreme negative value at the "innermost" parallel. We note and will heavily exploit the observation that the largest positive curvature at any point is always the value of k_1.

We will adopt the following notation. A point on the torus is denoted as P_i; it is considered to be vector in three space. We will abbreviate the vector $P_i - P_j$ as $P_{i,j}$. Likewise, we will refer to the unit normal vector of the surface at P_i as N_i, and a similar comment applies to vectors of the form $N_{i,j}$. Note that the values of the points and the normals are available as input data, measured in the coordinate system of the imaging apparatus. We will often refer to the "translated" points $T_i = P_i - rN_i$; if the r is the true minor radius, then $T_i = S_i$, that is a point on the torus spine.

2. Computing Minor Radius

It is not hard to show that when computing the parameters of a two-dimensional surface in three space, the information that a point is on the surface can be used to fix one parameter, and the information that a given vector is the normal to the surface at a point can be used to fix two parameters. (Analogous statements can be made about curvature, but we avoid curvature information because of its intrinsic susceptibility to noise.) Thus, a sphere is uniquely determined by four points, or, more simply, by two points and the normal at one of them. Thus, two points and their normals are insufficient to uniquely determine a torus, since they fix only six parameters. However, three points and their normals in general overdetermine the torus in the following way. There must exist a local space circle that serves as the torus spine, and it must be equidistant from all three points (this common distance is the value of r). But, in addition, this space circle must have the property that at the point of closest approach to each of the P_i, the tangent to the space circle must be perpendicular to the corresponding N_i. It is the need for some means to guarantee this tangent property that leads to the following observations and constructions.

We select a distinguished point P_i; without loss of generality assume that it is the point P_1. We then construct all the space circles through the translated point T_1 whose tangents at T_1 are perpendicular to N_1, and that also pass through a second translated point, call it T_2. There is a one degree of freedom family of such circles, and they all lie on the surface of a sphere. That such a construction is possible can be seen both geometrically and algebraically; it is also a special case of the theorem of Meusnier. (Meusnier's theorem states that if a set of planes are drawn through a tangent to a surface in a non-zero curvature direction, then the osculating circles of the intersections with the surface lie upon a sphere [Struik, 1961].)

Geometrically, we consider T_1 to be the south pole of a sphere whose north-south axis is collinear with N_1. The size of the sphere is determined by T_2. Call this sphere the supporting sphere. Any circle (great or little) that passes through T_1 and T_2 now also has its local tangent at T_1 perpendicular to N_1. Algebraically, we can look for the size of the support sphere, s; since we know that the center is constrained to lie along the direction of the N_1, the center is given by $C = T_1 + sN_1$. Both T_1 and T_2 lie on a common sphere if their distances from the center are equal. This is captured by equating the norms of their relative position vectors, giving in vector form the equation of the plane of their perpendicular bisectors: $|C-T_1| = |C-T_2|$. Expanding, we find that $s_2(r)$, that is, the size of the support sphere needed to accommodate T_2 ,is a function of r.

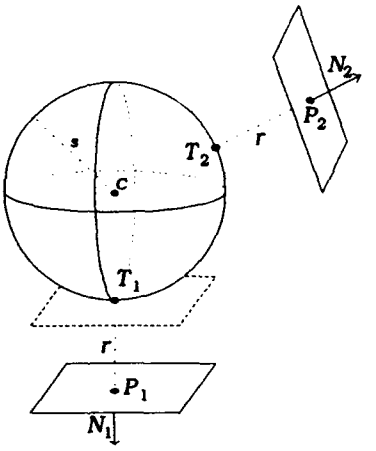


Fig. 2: Minor r extraction geometry.

$$s_2(r) = \frac{2r\,N_2 \cdot P_{1,2} - P_{1,2} \cdot P_{1,2}}{2r\,(1 - N_1 \cdot N_2) + N_1 \cdot P_{1,2}}$$

Note, however, that the local tangent at T_2 is not necessarily perpendicular to N_2. Worse, T_3 may not even lie on the support sphere at all.

We remedy this second defect first, by constructing with T_1 and T_3 the same sort of support sphere as was constructed with T_1 and T_2. What results is a symmetric relationship,

$$s_3(r) = \frac{2r\,N_3 \cdot P_{1,3} - P_{1,3} \cdot P_{1,3}}{2r\,(1 - N_1 \cdot N_3) + N_1 \cdot P_{1,3}}$$

which expresses the size of the support sphere which would not only accommodate T_3, but which would also allow us to draw a one-dimensional family of circles through T_1 and T_3 whose local tangents at T_1 are perpendicular N_1.

Now note that if $s_2(r)=s_3(r)$, for whatever r, the support spheres must be coincident, since they are both defined relative to P_1, and all three T_i lie upon it. Basically, equating $s_2(r)=s_3(r)$ will find those r that allow a solution to the now overdetermined problem of finding a sphere that passes through three points and attains a specified normal at a given point (here, the south pole). Equating $s_2(r)=s_3(r)$ gives a quadratic in r; $Ar^2+Br+C = 0$, where

$$A = 2\,N_2 \cdot P_{1,2}\,(1 - N_1 \cdot N_3) - 2\,N_3 \cdot P_{1,3}\,(1 - N_1 \cdot N_2)$$

$$B = 2\,N_2 \cdot P_{1,2}\,N_1 \cdot P_{1,3} \; - \; 2\,N_3 \cdot P_{1,3}\,N_1 \cdot P_{1,2} - P_{1,2} \cdot P_{1,2}\,(1 - N_1 \cdot N_3) \; + \; P_{1,3} \cdot P_{1,3}\,(1 - N_1 \cdot N_2)$$

$$C = -P_{1,2} \cdot P_{1,2}\,N_1 \cdot P_{1,3} + P_{1,3} \cdot P_{1,3}\,N_1 \cdot P_{1,2}$$

This equation will have at most two solutions for the value of r, and they are easily obtainable by the quadratic formula. Since they are based on T_1 being the south pole, call them $r_{1,1}$ and $r_{1,2}$. We still have to address the need for T_2 and T_3 to satisfy their tangent condition, however. We do this in a completely symmetric fashion, by finding two more support sphere systems, first by considering T_2 a south pole, and then T_3. The quadratic equations in r that result are derived by inspection by permuting the indices of the P_i and N_i.

From T_2 as south pole we get two more candidates for the value of r, $r_{2,1}$ and $r_{2,2}$, and from T_3 we get $r_{3,1}$ and $r_{3,2}$. This would appear to call for the calculation of 24 inner products, but it is easy to show that any two of these three quadratics have four inner products in common; thus, there is a total of only 12 innerproducts.

We now have three pairs of candidate r values. Let us define a consensus r to be any $r_{i,j}$ that satisfies

$$(2a) \qquad (r_{1,1} \text{or } r_{1,2}) = (r_{2,1} \text{or } r_{2,2}) = (r_{3,1} \text{or } r_{3,2})$$

There may be zero, one, or two consensus r. They can be accumulated and filtered in the usual Hough way using a one-dimensional parameter space. We note several properties of this consensus algorithm.

The values of r are derived by independently solving three quadratics in 15 image observables (three for each point P_i, and two for each unit normal N_i). The virtue of this method is that despite the non-linearities that would result when variables are eliminated, the method does not require any iterative root finding. Nor does it suffer from the attendant problems of choosing starting values, guaranteeing convergence, or tracking multiple roots.

Enlisting the aid of IBM's proprietary symbolic math system Scratchpad III [Jenks *et al.*, 1986], we were able to show that the transform always produced one consensus value for r, corresponding to the true minor radius of the torus, and that if a second consensus existed, it was always easily distinguishable by its sign and/or magnitude from the true consensus r, which is always the smallest positive radius of curvature.

Further, the analysis suggested several intriguing examples of what should probably be called anti-tori. The most straightforward example is what happens when two of the data points are on a meridian, and the third is on the innermost parallel. The transform properly returns two consensus: one is r, and the other is $-1/(a-r)$, the value of k_2 at the "innermost" parallel. What the transform "sees" is an anti-torus whose thickness is equal to the torus' hole, and whose hole is equal to the torus' thickness. That is, it interprets the data to be lying on the negative image of an anti-torus, whose axis is perpendicular to the given torus, with the torus and anti-torus interlocking: much like confusing the impression in plaster of a face with the face itself.

3. Computing Orientation (and Major Radius)

Following the principle of least variability, we next recover orientation by examining the spine points produced by our candidate r. Orientation is a naturally bounded quantity, and is easily represented as a point on the surface of the upper Gaussian hemisphere (mathematically, on P^2).

The approach is straight forward: given a candidate r, we form the spine points $S_i = P_i - rN_i$ from the pixels supporting r, and determine the orientation of the plane on which they lie. The direction of the plane's normal is easy to compute: simply take the normalized unit vector of the following cross product (or any of its variants obtained by permuting the S_i):

$$(S_1 - S_2) \times (S_3 - S_2)$$

However, this orientation is ill-conditioned when the spine points are nearly co-linear, which occurs when a is large. Using derivatives, it is not hard to show that small changes of S_2 in a direction perpendicular to this plane have an effect on the plane's orientation that is roughly proportional to the major radius, a. This suggests that the computation for orientation should be accompanied by the computation of a, so that orientation can be "weighted" inversely by a somehow.

Rather than use a as a pure parameter space voting weight, we note the following. The value of a eventually has to be recovered anyway, and if it disappears into a weighted sum it is not recoverable. Instead, we scale the unit orientation vector by dividing it by a. The parameter space now becomes the interior of the upper Gaussian hemisphere, which is three dimensional, and the values of a are recoverable simply by taking each vector's length. More importantly, orientations accompanied by high a cluster near the origin, where they are easy to detect and remove. Conversely, other orientations receive parameter space representation proportional to their certainty, since they are proportionately distant from the hemisphere center.

This parameter space has two other advantages. First, it uses the interior of what would otherwise be a very inefficient use of three-space, thus obviating the need to cleverly tessellate the surface of the Gaussian sphere. Secondly, points that cluster at its origin can immediately be considered evidence of cylindrical objects in the image (which have no orientation and infinite a).

Since the orientation and a spaces are now combined, the computation of the resulting parameter point (the scaled orientation vector) can also be combined; it is not necessary to actually plot the spine circle or calculate its center. By means of a three-space geometric construction related to the planar "Law of Sines" for circles, it can be shown that the scaled vector is directly given by the following (or by any of its variants obtained by permuting the S_i):

$$\frac{2W_1 \times W_3}{|W_1||W_2||W_3|}$$

where

$$W_1=[S_3-S_2],\ W_2=[S_1-S_3],\ \text{and}\ W_3=[S_1-S_2].$$

4. Computing Center

What remains are the three most difficult parameters, the center parameters, which are potentially unbounded and highly sensitive to the value of a. Local tori with small curvature have distant centers which are difficult to compute accurately. Since triangulation error of the center is also roughly proportional to a, this final space computes and accumulates vectors to the center point, scaled again by a. In effect, this measures each center in units of major radius, thus relative error is nearly constant. It is not hard to show that this space is now bounded, since for local tori with large a, their center must be about a units from the image origin, otherwise they would not fall within the image. Thus, most tori have centers within one unit from the origin. The upper limit of this space is determined by small tori at the image edge; since a is bounded by r, and r is bounded by physical considerations, this limit is calculable directly.

We note that although many tori might map to the same scaled vector, they have already been classified and separated by the value of a in the prior parameter space. Thus, the nesting of these spaces is critical both for adjustment to error, and for disambiguation of results.

The computation is straightforward but a bit messy (however, it is again invariant to the permutation of the S_i):

$$C = \frac{aS_1+bS_2+cS_3}{g}$$

where

$$g=D_1D_3-DD;\ a=D_1D_3-DD_1;\ b=DD_2;\ c=D_1D_3-DD_3$$

and

$$D_1=W_1 \cdot W_1;\ D_2=W_2 \cdot W_2;\ D_3=W_3 \cdot W_3;\ D=W_1 \cdot W_3.$$

5. System Considerations

In practice, the above method relies on the ability to pick "good" triples of points P_i. Additionally, the performance and behavior of the method can be enhanced by simple "sanity checks" on computed intermediate results. Lastly, the orientation space can be improved by a judicious choice of "offset" for the scaled orientation vector. We handle these in turn.

Picking good triples is determined by enforcing a minimum and maximum distance between image points P_i; we call this distance the "radius of coherence" (ROC). If the ROC is too small, accuracy suffers due to small triangulation baseline; too large, and most triples do not lie on the same torus. For the minor radius space, it can be set in accordance with the expected range of minor radii, which can be determined from the imaging parameters. Empirically, we have found that a wide range between minimum and maximum ROC works well for minor r. The two other spaces have similar considerations, however the range between minimum and maximum ROC must be rather narrow or the system becomes overwhelmed with noise hypotheses. Fortunately, experience shows that a narrow range of ROC about r works very well for a large range of torus shapes.

Sanity checks are inexpensive checks on the data before it is used in a parameter transform. Our sanity checks are of the following form. In computing r, no points in a planar neighborhood are used; they are easy to detect, since they have very small and equal principal curvatures. Nor are points near a depth discontinuity used, as the surface approximations become inaccurate there. Further, degenerate quadratics (with $A=0$ or with imaginary roots) are ignored . In computing orientations and centers, spine points within a pixel of each other are ignored. These checks take a minute percentage of the computation times, but reduce the noise in parameter space dramatically.

In practice, the scaling of orientation vectors by $1/a$ results in vectors too tightly clustered around the origin. This is because any torus large enough to be seen in the image as a torus will have a major radius of at least eight pixels, approximately. Thus, all the activity in the space happens in a hemisphere of radius $1/8$; this uses less than 1% of the space. If a lower bound on a is known (and it usually can be approximated), then the scaling function should be of the form $b/(a+c)$, with c serving to shift small values of a closer to the surface of the hemisphere. Heuristic choices of constants based on expected torus and image sizes can be selected; one good one maps the smallest torus onto the surface of the hemisphere, and the largest torus that can be fully seen (a=64, assuming a range image of size 256) into the midpoint of the hemisphere; the scaling function becomes $56/(a+48)$.

6. System Description

We briefly survey the complete system of which this transform is a part, highlighting those aspects most germane to our results. A more complete description can be found in [Kjeldsen *et al.*, 1989].

Recognition is structured as a hierarchy of layered and concurrent parameter transforms [Ballard, 1981][Sabbah, 1985]. Each transform examines input data or previously established features and accumulates evidence for new feature hypotheses in an associated parameter space. Compatibility relations accumulate evidence for or against a hypothesis on the basis of peer hypotheses. A large number of hypotheses are typically generated. The evidence for and against each is integrated using an iterative refinement process in a dynamically constructed constraint satisfaction network [Feldman & Ballard, 1981].

Each parameter space is instantiated as a subnetwork where nodes correspond to hypotheses. The links in the network are (1) bottom-up connections, representing support from input data, and (2) lateral links between hypotheses. The former links can be thought of as votes generated by the parameter transforms [Hough, 1962]. The latter links, generated by the compatibility relations, can be inhibitory, or excitatory. For example, surface hypotheses generated from the same pixels are connected by an inhibitory link, because they represent conflicting interpretations of the same data.

Evidence integration works as follows. Each node i computes an activation level u_i representing a confidence level in the corresponding feature. At each iterative step t, the activation level of a node, denoted by $u_i(t)$, is computed as

$$u_i(0)=0$$
$$u_i(t+1)=u_i(t)+I_i+\sum_j w_{ij}\,u_j(t)-D_i\,,$$

where I_i represents bottom-up support for feature i, and the summation embodies the collective inhibition and excitation of conflicting and cooperating hypotheses j. The weight factor w_{ij} is negative when hypothesis j conflicts with hypothesis i and positive when hypothesis j is supporting i. D_i is a decay term that suppresses spurious hypotheses with little support, and helps insure stability.

A unit "survives" iteration when it has sufficient u_i and insignificant inhibition. It is then passed to the next parameter transform in order to create hypotheses in higher-level spaces. Units also feed-back to their component features in lower-level spaces and to consistent hypotheses in parallel spaces. Thus, surviving features form stable coalitions which represent globally consistent interpretations of the scene.

Additional features are added to the system by defining parameter transforms and compatibility relations which work within a well defined I/O structure. These generally make use of techniques described in [Califano, 1988] and [Califano *et al.*, 1988]. Our first transform takes triples of data points and returns values of r on the basis of the consensus in Eq 2a. The second transform takes triples of the data points that support a surviving r hypothesis, and returns the parameters of the scaled vector of section 3. Finally data points supporting surviving orientation/major radii are used as described in section 4. The current system contains 21 parameter transforms for various features.

The lowest level of the system extract local features such as surface approximations or depth discontinuities from the data for use by the parameter transforms. The local features important to this work use bicubic interpolations to obtain the least mean square error fit to dense depth data, then computes directly from the polynomial coefficients of the approximation both the surface gradient vector, and the directions and amounts of principal surface curvature [Sabbah & Bolle, 1986]. Experiments comparing computed values with known true values indicate that over a wide range of imagery and circumstance the inaccuracy of the approximations is no more that 5% from the ideal.

7. Results

The torus extraction transforms have been run on about a dozen images. We will present two of the more interesting cases. The first is a depth map generated from a CSG representation of a padlock (figure 3). The image contains a single torus segment, as well as several other surfaces. The second image is an actual range image of a knotted length of cable (figure 4). The cable forms a continuously varying tube of constant cross section, which can be reasonably approximated as piecewise toroidal.

In both cases we used only a small percentage of the possible triples within the ROC; the percentage was large enough to give reasonable coverage but small enough to give acceptable running times. Additionally, hypotheses receiving votes from fewer than 20 triples were not instantiated. Since many noise hypotheses receive just a few votes, this pruning helped cut down on the time and memory needed to support them.

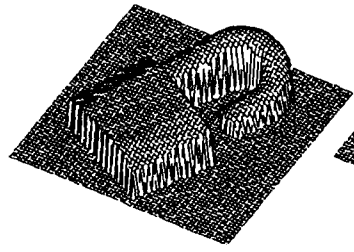


Fig. 3: Depth map of lock.

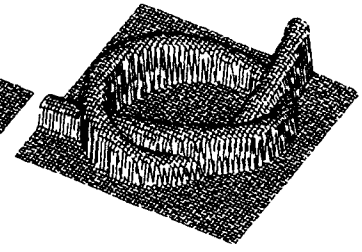


Fig. 4: Depth map of knot.

7.1. Lock

Figure 3 shows a 64x64 plot of the 256x256 depth map of the lock. Surface approximations were taken using a 5x5 window around each point. After the sanity checks, only the points hilighted in the dithered image in figure 5 were passed to the minor radius parameter transform. ROC was 9 to 10, and 50% of the possible triples were used. 10 hypotheses received votes from more than 20 triples, and so took part in iteration. After 11 iteration steps a single hypothesis of 16.5 for r survived, which corresponds closely to the apparent radius of the hasp.

Figure 6 shows the points which supported the winning r. To find the orientation/major radius, ROC was set to range from 15 to 16 (approximately r) and 50% of the triples formed from the spine points were used. 102 hypotheses received sufficient support to be instantiated.

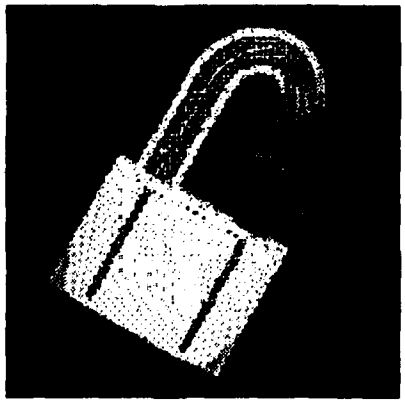


Fig. 5: Points voting for minor radius.

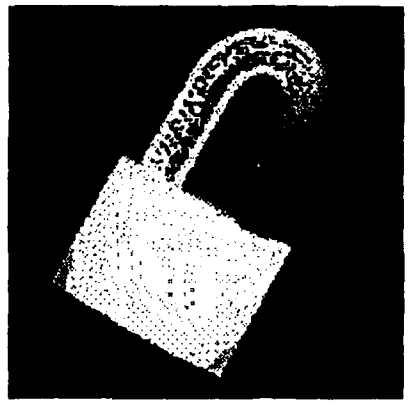


Fig. 6: Pixels supporting winning r.

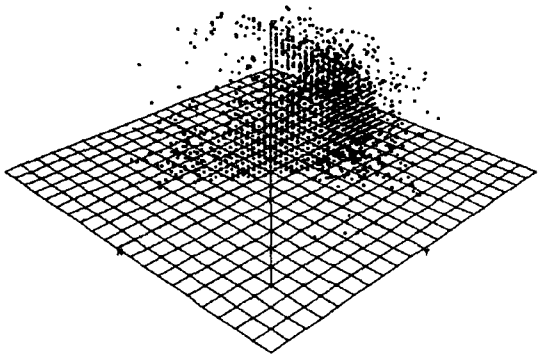


Fig. 7: Orientation hyotheses plotted in 3-space.

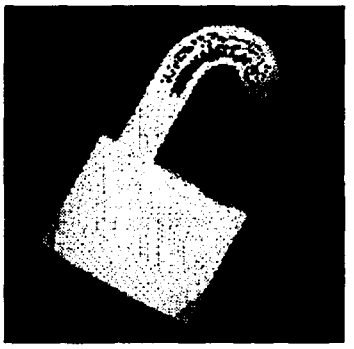


Fig. 8: Pixels voting for orientation 1.

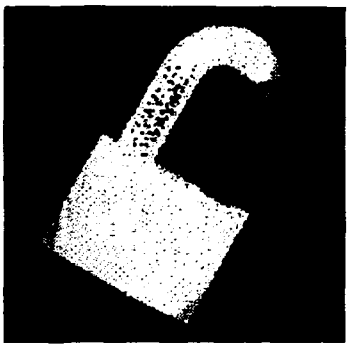


Fig. 9: Pixels voting for orientation 2.

Figure 7 shows the hypotheses receiving votes plotted in 3-space (it does not show the number of votes each received). You can make out the hemispherical outlines of the parameter space, as well as a distinct cluster of hypotheses in the -x +y +z quadrant. Two hypotheses survived after iteration. The points supporting them are shown in figures 8 and 9. The first captured the orientation of the torus forming the top of the hasp as closely as we were able to measure it ourselves. The major radius was captured as accurately as the parameter space resolution (bucket size) would allow (actual a=34, computed a=37). The second surviving orientation hypothesis "found" a very large diameter torus (a=523) on the straight segment of the hasp. As we mentioned in section 3, it is possible to eliminate hypotheses of very large radius, that is near-cylinders by discarding those very close to the origin of the parameter space. Since the length of the parameter vector of the torus on top of the hasp is .65, and the length of the parameter vector for the torus found on the cylinder was .1, the cutoff could easily be adjusted to eliminate such misinterpretations.

Both orientation hypotheses now voted into location space. The same ROC and hit rate were used. The cylindrical segment was not able to find any consistent location, and therefore created no hypotheses strong enough to be considered. The actual torus created 25 hypotheses. Only one survived iteration. The surviving center was (30 67.5 -30), with the actual center at approximately (31 66 -32). Thus, the system found the results to within one bucket of available resolution.

These runs were done on a Symbolics 3650. With the parameters set as described the entire recognition took roughly 70 minutes. Computing the surface approximations took 10 minutes. Generating the votes for minor r space took 30 minutes. Generating votes for orientation space took 25 minutes. Generating votes for location space took 3 minutes. In all, 30 iterations were needed to prune down the hypotheses. Total time for all iteration was roughly 3 minutes, including a rather elaborate trace of system status.

7.2. Knot

The second image is a range image of a knot of coax cable (figure 4) taken using a laser triangulation range finder [Technical Arts, 1986]. ROC was the range 4 to 5, hit rate was 50%. Figure 10 shows the points supporting the winning (and correct, as far as we can measure) minor radius hypothesis.

These points were passed thru the orientation/major radius parameter transform. Despite the lack of distinct peaks in the histogram of votes (figure 11), the iterative refinement was able to find three distinct clusters, representing competing hypotheses of orientation from different areas of the knot. The strongest hypothesis in each cluster survived iteration. The points supporting them are shown in figures 13, 14 and 15. The light coverage is due to the low hit ratios (10%) we had to use. The low hit rate also appears responsible for some regions of the knot not being covered by any hypotheses.

Figure 12 shows a histogram of the votes the three orientation hypotheses generated in location space. One peak was generated by the segment in fig.15, and the other peak is an overlap of the votes from both fig. 13 and fig. 14. The two locations which survived iteration correspond well to the center points we expected from those torus segments.

Running times were somewhat better than those for the lock test case, except for the time to generate votes for orientation hypotheses. Here the continuously varying curve created a very large number of noise hypotheses, and very wide peaks in the histogram of votes. With the 10% hit rate used

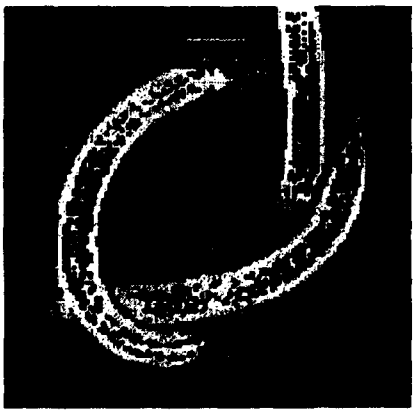


Fig. 10: Pixels supporting winning minor radius.

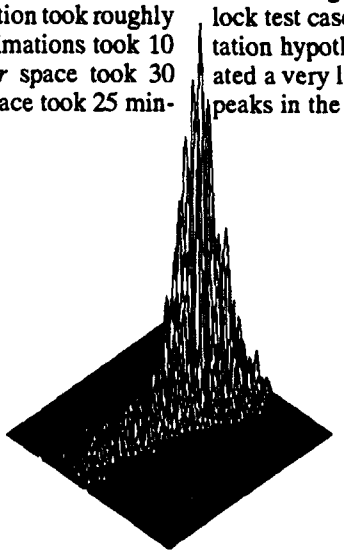


Fig. 11: Knot orientation votes / 2D projection.

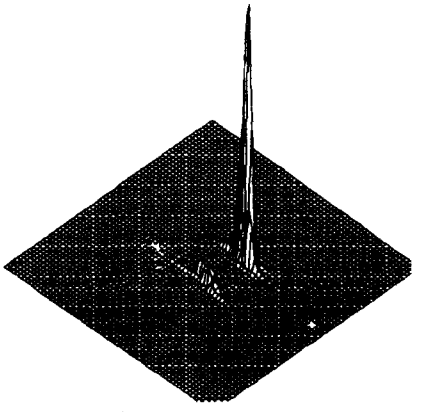


Fig. 12: Pixels supporting winning location.

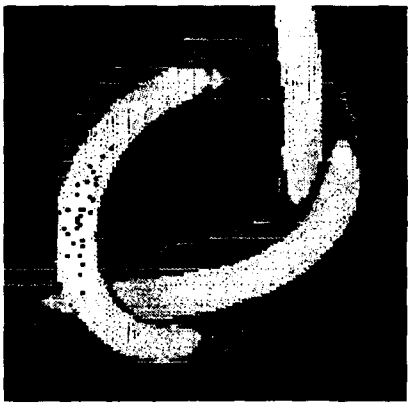


Fig. 13: Pixels supporting orientation hypothesis 1.

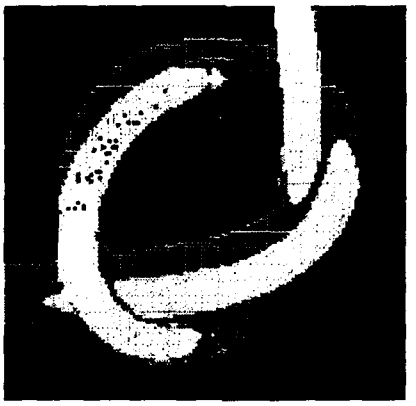


Fig. 14: Pixels supporting orientation hypothesis 2.

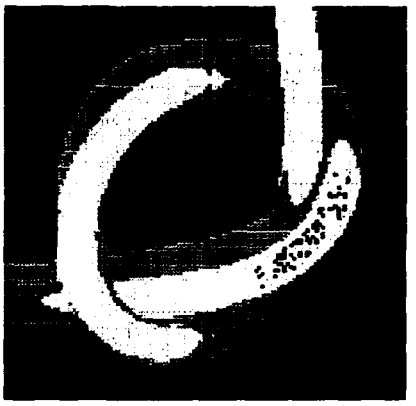


Fig. 15: Pixels supporting orientation hypothesis 3.

here, roughly 4000 hypotheses were created. Linking them took well over an hour. With somewhat higher hit rates, 21000 hypotheses were created. Many hours were needed to create and link them.

7.3. Discussion

These experiments demonstrate the ability to break a continuously varying curve into piecewise toroidal segments, to deal with non-toroidal surfaces and cylinderical segments. We hope that speeding the implementation and a future port to faster hardware will improve the coverage, but these results are very promising.

System parameters must currently be adjusted for each image. An initial setting generally gives either impossibly long running times, due to the number of hypotheses created, or poor coverage of points supporting the winning hypotheses. The latter makes recognition in subsequent parameter spaces difficult. ROC and hit rate can generally be adjusted for good coverage in reasonable time, but it takes several attempts. A faster implementation would allow us to use all triples and a wider ROC, especially in minor radius space. We believe this would almost eliminate the need to tune system parameters.

Acknowledgments

This work was performed while the first author was partially supported on sabbatical leave by IBM Hawthorne; additional support was provided by the Defense Advanced Research Projects Agency, contract N00039-84-C-0165. We thank Sanjaya Addanki, Ruud Bolle, Andrea Califano, and Rakesh Mohan for their generosity and insight. We especially are grateful to another visitor, Russell W. Taylor, for many useful discussions over donuts, and anti-donuts.

References

[Ballard, 1981] D.H. Ballard, "Parameter nets: A theory of low level vision," in *Proc. 7th Int. Joint Conf. On Artificial Intell.*, Aug. 1981, pp. 1068-1078.

[Califano, 1988] A. Califano, "Feature recognition using correlated information contained in multiple neighborhoods," in *Proc. 7th Nat. Conf. on Artificial Intell.*, July 1988, pp. 831-836.

[Califano *et al.*, 1988] A. Califano, R.M. Bolle, and R.W. Taylor, "Generalized neighborhoods: A new approach to complex feature extraction," *IEEE Conf. on Comp. Vision and Pattern Recognition*, Nov. 1988.

[DoCarmo, 1976] M.P. DoCarmo, *Differential geometry of curves and surfaces*. New Jersey: Prentice-Hall, 1976.

[Feldman & Ballard, 1981] J.A. Feldman and D.H. Ballard, "Connectionist models and their properties," *Cognitive Science*, Vol. 6, 1981, pp. 205-254.

[Hough, 1962] P.V.C. Hough, Methods and Means for Recognizing Complex Patterns, U.S. Patent 3069654, 1962.

[Jenks *et al.*, 1986] R.D. Jenks, R.S. Sutor, and S.M. Watt, "Scratchpad II: An abstract datatype system for mathematical computation," IBM Tech. Rep. RC12327, Nov. 1986.

[Kender & Kjeldsen, 1991] J. R. Kender and R. Kjeldsen, "On Seeing Spaghetti: A Radius-finding Transform for Flexible Extruded Objects," IBM Tech Rep. RC16576, Feb. 1991.

[Kjeldsen *et al.*, 1989] R. Kjeldsen, R.M. Bolle, A. Califano, and R.W. Taylor, "A Homogeneous Framework for Visual Recognition," in *Proc. 11th Intl. Joint Conf. on Artificial Intell.*, August 1989, pp. 1578-1584.

[Sabbah, 1985] D. Sabbah, "Computing with connections in visual recognition of origami objects," *Cognitive Science*, Vol. 9, No. 1, Jan.-March 1985, pp. 25-50.

[Sabbah & Bolle, 1986] D. Sabbah and R.M. Bolle, "Extraction of surface parameters from depth maps viewing planes and quadrics of revolution," in *Proc. SPIE Conf. Intell. Robots and Comp. Vision*, Oct. 1986, pp. 222-232.

[Struik, 1961] D.J. Struik, *Lectures on Classical Differential Geometry*, Addison-Wesley, 1961.

[Technical Arts, 1986] Technical Arts Corporation, *100X 3D Scanner: User's manual & application programming guide*. Redmond, WA: 1986.

Shape from Focus System for Rough Surfaces

Shree K. Nayar

Department of Computer Science, Columbia University, New York, N.Y. 10027

Abstract

All surfaces encountered in practice appear rough at some level of detail. Existing shape recovery methods are effective when applied to smooth diffuse surfaces but produce sparse shape information when applied to rough surfaces. Images of rough surfaces are characterized by high frequency intensity variations, and it is difficult to perceive the shapes of these surfaces from their images. The shape-from-focus method described in this paper uses different focus levels to obtain a sequence of object images. The sum-modified-Laplacian (SML) operator is developed to compute local measures of the quality of image focus. The SML operator is applied to the image sequence, and the focus measures obtained at each image point are used to compute local depth estimates. We present two algorithms for depth estimation. The first algorithm simply looks for the focus level that maximizes the focus measure at each image point. The second algorithm uses a model to interpolate the focus measures to obtain more accurate depth estimates. The algorithms were implemented and tested on surfaces of different roughness and reflectance properties. We conclude with a description a fully automated shape-from-focus system that has been applied to industrial samples.

1 Introduction

The advancement of three-dimensional machine vision is largely dependent on the development of efficient and reliable shape extraction methods. Shape extraction, in turn, requires a clear understanding of surface reflectance and image formation. Several extraction methods, for diffuse and specular surfaces, have been developed in the past. The extraction problem associated with *rough surfaces*, however, has not received sufficient attention. All surfaces encountered in practice are rough at some level detail. At that level, they exhibit high-frequency spatial surface variations that are often random in nature. In many vision applications, the spatial surface variations are comparable in dimensions to the resolution of the imaging system. Image intensities produced by such surfaces vary in an unpredictable manner from one sensor element (pixel) to the next. Hence, it is difficult to obtain dense and accurate surface shape information by using existing passive or active sensing techniques such as stereo, shape from shading, and structured light. Therefore, a practical and reliable solution to this rather difficult extraction problem is desirable. In this paper, we develop a shape extraction technique that uses focus analysis to recover dense depth maps of rough textured surfaces.

1.1 Background

Previously, focus analysis has been used to automatically focus imaging systems or to obtain *sparse* depth information from the observed scene. Horn [1] proposed focusing imaging systems by using the Fourier transform and analyzing the frequency spectrum of the image. Tenenbaum [2] developed the gradient magnitude maximization method that uses the sharpness of edges to optimize focus quality. A modification to this approach was later proposed by Jarvis [3]. He formulated the sum-modulus-difference as the sum of the first intensity differences between neighboring pixels along a scan-line and used it as a measure of focus quality. Several automatically focusing algorithms were implemented and tested by Schlag et.al. [4].

More recently, Krotkov [5][6] evaluated and compared the performance of different focus criterion functions. Krotkov also proposed a method to estimate the depth of an image area. Pentland [7] suggested estimating the depth of image points by evaluating image blur due to defocusing. A similar approach was applied to edge points by Grossmann [8]. Darrell and Wohn [9] have developed a depth from focus method that obtains an image sequence of a scene by varying the focus level, and uses Laplacian and Gaussian pyramids to calculate depth. Subbarao [10] suggests the change of intrinsic camera parameters to recover the depth map of a scene. Ohta et.al. [11] and Kaneda et.al. [12] have used images corresponding to different focus levels to obtain a single image of high focus quality.

1.2 Shape from Focus

In this paper, we develop a shape-from-focus method. In contrast to previous work in this area, we avoid the following approaches.

- We do not attempt to estimate depth from a pair of images by evaluating local estimates of the blurring function. The accuracy of such a method is greatly dependent on the blurring model used. The models used thus far are only approximations to the actual physical-optics model and therefore do not ensure high quality results.
- We do not apply our method to general scenes. Depth estimation based on focus analysis relies on the presence of high frequency brightness variation in the scene. General scenes often have areas with little or no brightness variation. For this reason, experiments in the past have only produced sparse depth information.

Here, we restrict ourselves to visibly rough surfaces that produce textured images with high frequency intensity variations. We review the image formation process and show that a defocused imaging system plays the role of a low-pass filter. The shape-from-focus method moves the unknown object with respect to the imaging system and obtains a sequence of images that correspond to different levels of object focus. The sum-modified-Laplacian (SML) focus operator is developed to measure the relative degree of focus between images. The operator is applied to the image sequence to obtain a set of focus measures at each image point. A model is used to describe focus measure variations due to defocusing. This model is used to interpolate between a finite number of focus measures to obtain accurate depth estimates. Experimental results indicate that the method is capable of extracting dense and accurate shape information from a few images of the object. The results demonstrate appreciable invariance to surface roughness and reflectance.

1.3 An Automated System

We have implemented a fully automated shape-from-focus system. The system is applicable to objects that are a few hundred microns in size. An optical microscope is used to image the object. The translation stage of the microscope is motorized to enable the controlled movement of the unknown object through the plane of focus of the imaging system. The stage is translated in increments and for each new position an image of the object is obtained. The image sequence is processed on a workstation. Two results are produced by the recovery algorithm. The first is a focused image of the object that is reconstructed from the sequence of partially focused images. The second is a depth map of the object surface.

The automated system has been applied to a variety of surfaces. Our initial results were obtained using industrial samples such as via-hole fillings on a ceramic circuit boards. We are currently using the system to recover the shapes and focused images of a variety of biological samples, including micro-organisms and chromosomes.

2 Visibly Rough Surfaces

In the study of reflection, a rough surface is defined as one whose smallest spatial variations have dimensions that are much larger than the wavelength of the incident electromagnetic wave. This is the concept of optical roughness. In this paper, we introduce the notion of *visible roughness*; a surface is visibly rough if the dimensions of its spatial variations are comparable to the viewing area of individual elements (e.g. pixels) of the sensor (e.g. camera) used to observe the surface. The surface shown in Fig.1 is comprised of a large number of facets[1]. While the surface appears to have a smoothly varying global shape, $z(x, y)$, the orientation α of individual facets may deviate considerably from the mean surface orientation in the facet vicinity. Although facet orientations are dependent on the global shape of the surface and on the orientations of neighboring facets, they often exhibit some degree of randomness.

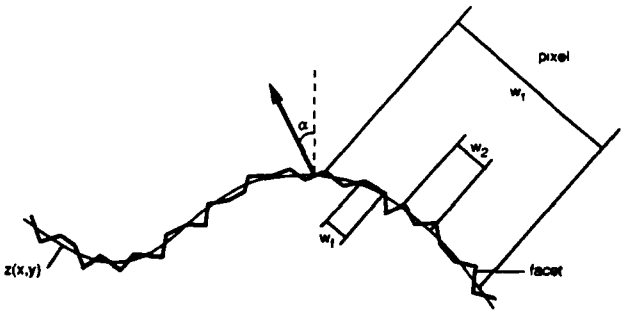


Figure 1: Surface roughness and sensor resolution.

Now let us consider the image of a rough surface obtained using a finite resolution sensor. The number of facets that contribute to the image irradiance at a pixel location depends on the magnification of the optical system used to project the surface onto the image plane of the sensor. We define two levels of magnification; *multi-facet level* and *facet level*. At the multi-facet level, the pixel width w is very large compared to the facet size w_f (e.g. $w = w_f$). In this case, the surface patch projected on a pixel may be modeled by assigning it a mean orientation value and a roughness which is determined by the probability distribution of its facet orientations [13] [14]. The pixel intensities are continuous functions of the angle of incident light and can be expressed as a linear combination of the diffuse lobe and specular lobe components [14], the relative strengths of the two components depending on the reflectance properties of the facets.

For facet level magnification, on the other hand, the pixel width w is comparable to the facet width w_f (e.g. w

[1]No assumptions are made regarding the size of the facets. Hence, these facets may or may not represent the *micro-facets* defined in [13], [14].

$= w_2$), and only one or few facets are viewed by each pixel. As a result of the randomness in facet orientations, image intensity values are expected to vary drastically and unpredictably from one pixel to the next. This is true for both specular as well as diffuse facets as the radiance of both are dependent on the angle of incident light. Therefore, at the facet level, a surface produces images that are rich in *texture*[2] and we say that the surface is *visibly rough*. Why then do we use facet level measurements when multi-facet level measurements will provide us with image intensities that can perhaps be used to recover shape information? In many practical situations, the desirable resolution of shape information is unobtainable at the multi-facet level.

3 Focused and Defocused Images

In this section, we briefly review the image formation process and describe defocused images as processed versions of focused images. Fig.2 shows the basic image formation geometry. All light rays that are radiated by the object point P and intercepted by the lens are refracted by the lens to converge at the point Q on the image plane. The relationship between the object distance o, focal distance of the lens f, and the image distance i, is given by the Gaussian lens law:

$$\frac{1}{o} + \frac{1}{i} = \frac{1}{f}. \tag{1}$$

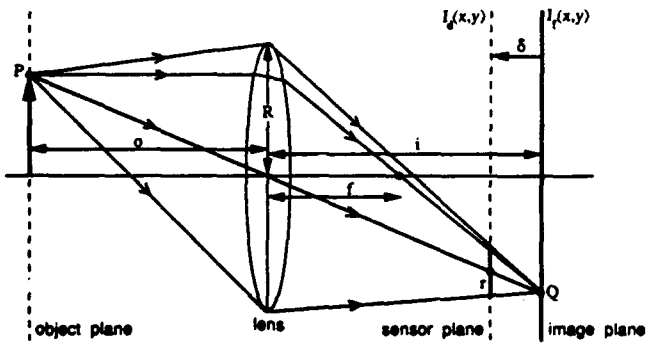


Figure 2: Formation of focused and defocused images.

Each point on the object plane is projected onto a single point on the image plane, thus causing a clear or *focused* image $I_f(x,y)$ to be formed on the image plane. If, however, the sensor plane does not coincide with the image plane and is displaced from it by a distance δ, the energy received from the object by the lens is distributed over a circular[3] patch on the sensor plane. Fig.2 may be used to establish the relationship between the radius r of the circular patch and the sensor displacement δ. From Fig.2 we find that:

$$r = \frac{\delta R}{i} \tag{2}$$

where R is the radius of the lens. It is also possible to convince oneself that the radius r of the circular patch is independent of P's location on the object plane. The distribution of light energy over the circular patch, or the *blurring function*, can be modeled using physical optics [17]. Very often, a two-dimensional Gaussian function is used to approximate the physical model [7]. Then, the blurred or *defocused* image $I_d(x,y)$ formed on the sensor plane can be described as the result of convolving the focused image $I_f(x,y)$ with the blurring function $h(x,y)$:

$$I_d(x,y) = h(x,y) * I_f(x,y) \tag{3}$$

where:

$$h(x,y) = \frac{1}{2\pi\sigma_h^2} e^{-\frac{x^2+y^2}{2\sigma_h^2}} \tag{4}$$

where σ_h, the *spread parameter*, is assumed to be proportional to the radius r [7]. The constant of proportionality is dependent on the optics, sampling, etc. We will see shortly that the value of this constant is not important in our approach. Note that defocusing is observed for both positive and negative sensor displacements.

Now let us analyze the defocusing process in the frequency domain (u,v). If $I_F(u,v)$, $H(u,v)$, and $I_D(u,v)$ are the Fourier transforms of $I_f(x,y)$, $h(x,y)$, and $I_d(x,y)$, respectively, we can express eq. 3 as:

$$I_D(u,v) = H(u,v) . I_F(u,v) \tag{5}$$

where:

$$H(u,v) = e^{-\frac{u^2+v^2}{2}\sigma_h^2} \tag{6}$$

We see that $H(u,v)$ allows low frequencies to pass while it attenuates the high frequencies in the focused image. Furthermore, as the sensor displacement δ increases, the defocusing radius r increases, and the spread parameter σ_h increases. Hence defocusing is a *low-pass* filtering process where the bandwidth decreases with increase in defocusing.

From Fig.2, it is seen that a defocused image of the object can be obtained in three ways: by displacing the sensor with respect to the image plane, by moving the lens, or by moving the object with respect to the object plane. Moving the lens or sensor plane with respect to one another causes the following problems:

- The magnification of the system varies, thereby causing the image coordinates of focused points on the object to change.

[2]There are many notions of what is meant by the term texture. Here, we define texture as a noticeable fluctuation in the intensities of neighboring image pixels [18]. The textures produced by rough surfaces may be periodic, nearly periodic, or random. No assumptions are made regarding the type of texture.

[3]The shape of the patch also depends on the shape of the aperture of the imaging sy em. We are assuming the aperture to be circular.

- The area on the sensor plane over which light energy is distributed varies, thereby causing a variation in image brightness.

These effects are described in detail by Willson and Shafer [20]. In order to overcome these problems, we propose to vary the degree of focus by moving the object[4] with respect to a fixed configuration of the optical system and sensor (Fig. 3). This approach ensures that as the object passes through the plane S, surface points that lie on S are perfectly focused onto the image plane with the same magnification. In other words, as the object moves, the magnification of imaging system can be assumed to be constant for the image areas that are perfectly focused.

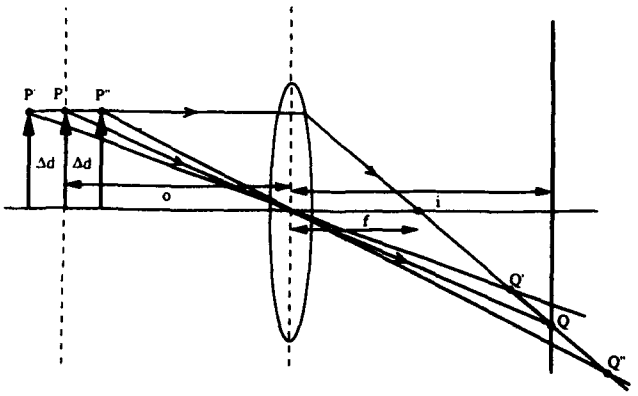


Figure 3: Effect of object displacement on magnification.

However, from Fig. 3 we see that points that lie outside of the plane S will be projected onto the image plane with different magnifications. In fact, the magnification of the defocused object points will depend on their distance from the plane S. Note that for small displacements Δd, magnification may be assumed to be constant. We will use this assumption while developing the depth estimation algorithm.

4 Shape from Focus: An Overview

The shape-from-focus method is based on the observations made in the previous sections.

- At facet level magnification, rough surfaces produce images that are rich in texture.
- A defocused optical system plays the role of a low-pass filter.

Fig.4 shows a rough surface of unknown shape placed on a translational stage. The reference plane shown corresponds to the initial position of the stage. The configuration of the optics and sensor defines a single plane, the "focused plane[5]," that is perfectly focused onto the sensor plane. The distance d_f between the focused and reference planes, and the displacement d of the stage with respect to the reference plane, are always known by measurement. Consider the surface element, s, that lies on the unknown surface, S. If the stage is moved towards the focused plane, the image of s will gradually increase in its degree of focus (high frequency content) and will be perfectly focused when s lies on the focused plane. Further movement of the element s will again increase the defocusing of its image. If we observe the image area corresponding to s and record the stage displacement $d = \overline{d}$ at the instant of maximum focus, we can compute the height d_s of s with respect to the stage as $d_s = d_f - \overline{d}$. In fact, we can use $\overline{d}$ to determine the distance of s from the focused plane, sensor plane, or any other coordinate system defined with respect to the imaging system. This procedure may be applied independently to all surface elements to obtain the shape of the entire surface S.

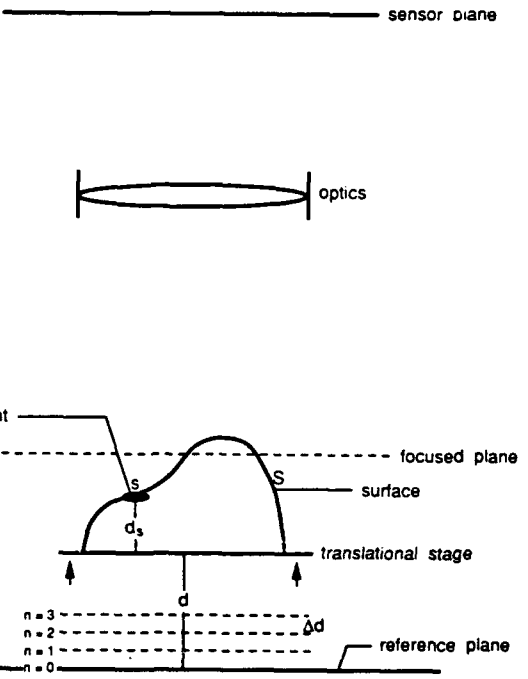


Figure 4: Shape from focus.

To automatically detect the instant of "best" focus, we will develop an image focus measure. In the above discussion, the stage motion and image acquisition were assumed to be continuous processes. In practice, however, it is not feasible to acquire and process such a large number of images in a reasonable amount of time. Therefore, we obtain only a finite number of images; the stage is moved in increments of Δd, and an image is obtained at each stage position ($d = n.\Delta d$). By studying the behavior of the focus measure, we develop an interpolation method that uses a small number of focus measures to compute accurate depth estimates. An important feature of the method

[4] Object movement is easily realized in industrial inspection applications.

[5] The focused plane is the same as the object plane defined in the previous section. A different term is introduced here as the object does not necessarily lie on the focused plane.

is its local nature; the depth estimate at an image point is computed only from focus measures recorded at that point. Consequently, the method can adapt well to variations in texture type and content over the object surface.

5 A Focus Measure Operator

To measure the quality of focus in a small image area, we develop a focus measure operator. The operator must respond to high frequency variations in image intensity, and ideally, must produce maximum response when the image area is perfectly focused. The high frequency content of an image area can be determined by using the Fourier transform. However, since Fourier transforms are expensive to compute and analyze without special purpose hardware, we seek an alternative method.

A few focus measure operators have been proposed and used in the past [5]. Generally, the objective has been to find an operator that behaves in a stable and robust manner over a variety of images such as images of indoor and outdoor scenes. Such an approach is essential while developing automatically focusing imaging systems that have to deal with general scenes. Bearing in mind that we are dealing with textured images, we develop an operator that is particularly well-suited to such images. In the next section we will evaluate the performance of our operator.

An interesting observation can be made regarding the application of focus measure operators. Eq. 3 relates a defocused image to a focused image using the blurring function. Assume that a focus measure operator $o(x,y)$ is applied (by convolution) to the defocused image $I_d(x,y)$. The result is a new image $r(x,y)$ that may be expressed as:

$$r(x,y) = o(x,y) * (h(x,y) * I_f(x,y)) \quad (7)$$

Since convolution is a linear operation, we can rewrite the above expression as:

$$r(x,y) = h(x,y) * (o(x,y) * I_f(x,y)) \quad (8)$$

Therefore, applying a focus measure operator to a defocused image is equivalent to defocusing a new image obtained by convolving the focused image with the focus measure operator. The focus measure operator only selects the frequencies in the focused image that will be attenuated due to defocusing. Since defocusing is a low-pass filtering process, its effects on the image are more pronounced and detectable if the image has strong high frequency content. An effective focus measure operator, therefore, must high-pass filter the image.

One way to high-pass filter an image is to determine its second derivative. For two-dimensional images, the *Laplacian* may be used:

$$\Delta^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \quad (9)$$

where $I(x,y)$ is the image intensity at the point (x,y). In frequency domain, applying the Laplacian $L(u,v)$ to the defocused image $I_D(u,v)$ (eq. 5) gives:

$$L(u,v) . H(u,v) . I_F(u,v) \quad (10)$$

where:

$$L(u,v) . H(u,v) = -(u^2 + v^2) . e^{-\frac{u^2+v^2}{2}\sigma_h^2} \quad (11)$$

Fig.5 shows the frequency distribution of $| L . H |$ as a function of the defocusing parameter σ_h. For any given frequency (u,v), $| L . H |$ varies as a Gaussian function of the defocusing parameter σ_h. In general, however, the result would depend on the frequency distribution of the imaged scene. Though our texture is random, it may be assumed to have a set of dominant frequencies. Then, loosely speaking, each frequency is attenuated by a Gaussian function in σ_h and its width is determined by the frequency. Therefore, the result of applying the Laplacian operator may be expressed as a sum of Gaussian functions in σ_h. The result is expected to be maximum when $\sigma_h = 0$, i.e. when the image is perfectly focused. Since the frequency distribution of the texture is random, the widths of the Gaussian functions are also random. Using central limit theorem, the result of applying the Laplacian operator to an image point may be assumed to be a Gaussian function of the defocusing parameter σ_h.

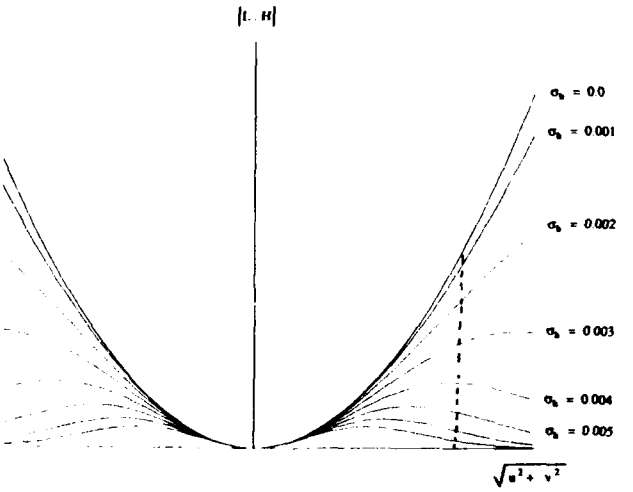


Figure 5: The effect of defocusing and second-order differentiation in frequency domain.

This general behavior is expected irrespective of the focus measure operator used. The focus measure operator only selects the frequencies that will play a dominant role in this process. Our experiments (section 5) as well as Krotkov's empirical evaluation of various focus measure operators [5] support the above argument. As seen in [5], image noise and magnification variations will of course degrade the performance any focus measure operator.

We note that in the case of the Laplacian the second derivatives in the x and y directions can have opposite signs and tend to cancel each other. An example of such an instance is illustrated in Fig.6; the partial derivatives are equal in magnitude but opposing in sign, i.e. $\Delta^2 I = 0$. In the case of textured images, similar instances may occur frequently and the Laplacian may at times behave in a unstable manner. We overcome this problem by defining the *modified Laplacian* as:

$$\Delta^2{}_M I = \left|\frac{\partial^2 I}{\partial x^2}\right| + \left|\frac{\partial^2 I}{\partial y^2}\right| \quad (12)$$

Note that the modified Laplacian is always greater or equal in magnitude to the Laplacian. In [15] the advantages of the above modification have been empirically demonstrated. However, the experiments described in [15] also indicate that the response of the modified Laplacian is slightly more stable but not very different from that of the Laplacian.

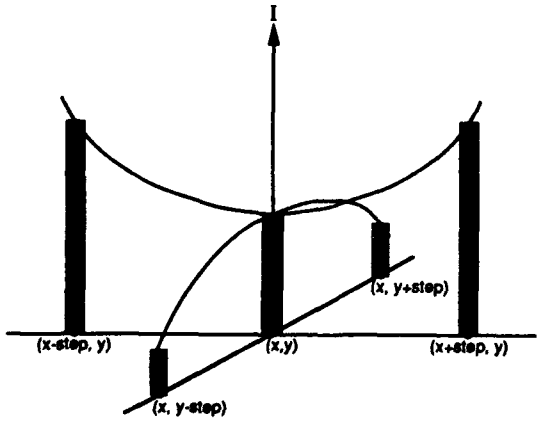


Figure 6: A texture instance with zero Laplacian value.

The discrete approximation to the Laplacian is usually a 3×3 operator. In order to accommodate for possible variations in the size of texture elements, we compute the partial derivatives by using a variable spacing (*step*) between the pixels used to compute the derivatives. Hence, the discrete approximation to the modified Laplacian is computed as:

$$\begin{aligned} ML(x,y) \;=\; & \quad (13) \\ & |\; 2I(x,y) - I(x-step,y) - I(x+step,y)\;| \\ + \; & |\; 2I(x,y) - I(x,y-step) - I(x,y+step)\;| \end{aligned}$$

Finally, the focus measure at a point (i,j) is computed as the sum of modified Laplacian values, in a small window around (i,j), that are greater than a threshold value:

$$F(i,j) = \sum_{x=i-N}^{i+N} \sum_{y=j-N}^{j+N} ML(x,y) \quad for\ ML(x,y) \geq T_1 \quad (14)$$

where the parameter N determines the window size used to compute the focus measure. In contrast to auto-focusing methods, we typically use a small window of size 3×3 or 5×5, i.e. $N = 1$ or $N = 2$. We shall refer to the above focus measure as the *sum-modified-Laplacian* (SML). Note that as a result of definition of the modified Laplacian and the use of the threshold T_1, the SML is not a linear operator and cannot be implement as a simple convolution. However, the SML can be computed using a straightforward algorithm.

6 Evaluating the Focus Measure

We evaluate the SML focus measure by analyzing its behavior as a function of the distance between the observed surface and the focused plane. A detailed description of the experimental set-up is given in a later section. In the following experiments, texture samples are attached to a translational stage (Fig.4) and the distance, d_s, from each sample to the stage is known by measurement. Images of the samples are obtained using a microscope and a 512×512 pixel CCD camera. The complete imaging system has a physical resolution of approximately $1\mu m$ per pixel width.

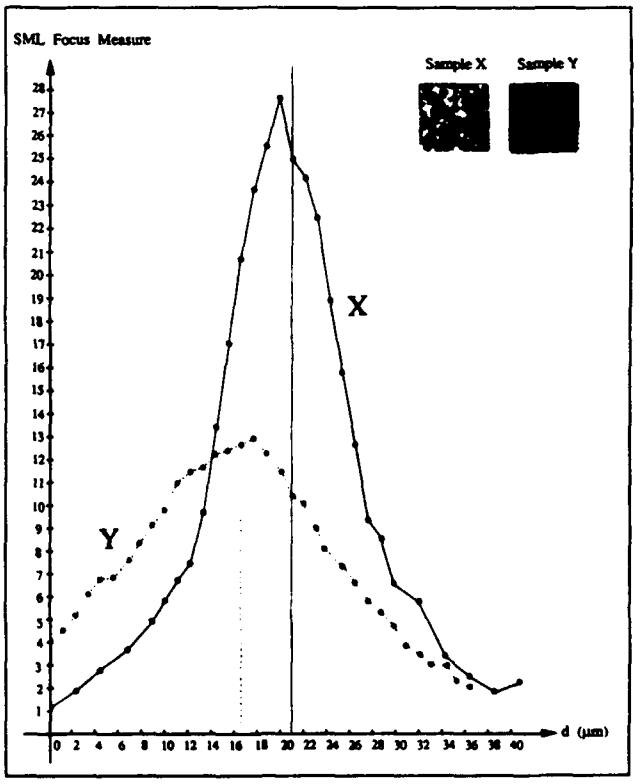


Figure 7: SML focus measure function computed for two texture samples.

In Fig.7, the focus measure functions of two samples are shown. Sample X has high texture content while sample Y

has relatively weaker texture. Both samples are made of a paste containing resin and tungsten particles. The variable size of the tungsten particles gives the surfaces a randomly textured appearance. For each sample, the stage is moved in increments (Δd) of $1\mu m$, an image of the sample is obtained, and the SML focus measure is computed using an evaluation window size of 10×10 pixels. The vertical lines in Fig.7 indicate the known initial distances (d_f - d_s) of the samples from the focused plane. The focus measures were computed using parameter values of *step* = 1 and T_1 = 7. No form of temporal filtering was used to reduce the effects of image noise, as we intend to use unfiltered focus measures to estimate the depth of surface points. Though the measure values are slightly noisy, they peak very close to the expected peak positions (vertical lines in Fig.7). We see that the focus measure function peaks sharply for the stronger texture and it peaks relatively slowly and with a lower peak value for the weaker texture. However, the sharpness of the focus measure function depends not only on the texture strength but also the depth of field of the imaging system. The depth of field, in turn, depends on the magnification and aperture size of the imaging system. We will assume that the depth of field is constant for all our experiments. Note that the focus measure functions for both samples have Gaussian-like distributions near their peak values. The fringes of the focus measure functions are less symmetric as the magnification can vary substantially from one fringe to the other.

Fig. 8 shows the focus measure computed as a function of the parameter *step* for the sample X shown in Fig.7. Once again, an evaluation window size of 10×10 and a threshold value of T_1 = 7 were used. We see that, for

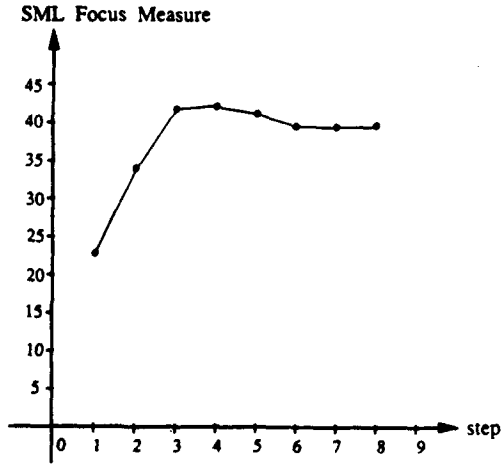


Figure 8: SML focus measure and the parameter *step*.

sample X, a maximum measure value is computed at *step* = 4. However, it may be noted that the effective size of the measure evaluation window increases with the *step* size. Since we are interested in local depth estimates, *step* values of 1 or 2 are usually used. Fig. 9 shows the effect of varying the threshold T_1, for both focused and defocused images of sample X. A good value for T_1 is one that produces a high measure value for the focused image and low measure values for defocused images. From Fig. 9, we see that, for sample X, T_1 = 7 appears to be a good choice. However, from a number of unreported experiments, we find that though the peak value of the focus measure function varies with the parameter values, the same parameter values may be used to obtain sharp, unimodal, focus measure functions for a large range of texture types and strengths.

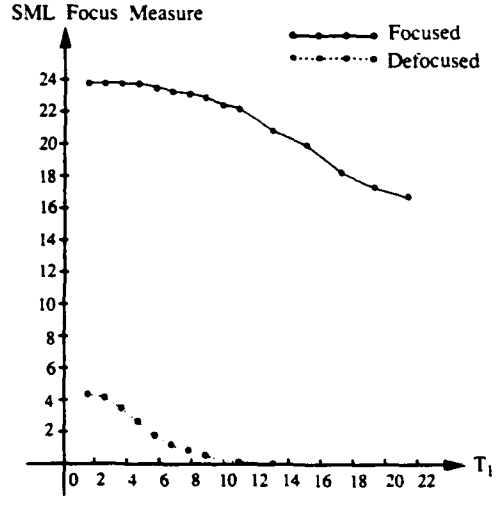


Figure 9: SML focus measure and the parameter T_1.

In [15], the SML focus measure has been compared with three other measures that have been previously used for auto-focusing: *Tenengrad*, *variance*, and *sum-Laplacian* (SL). These experiments indicate that, among these operators, the SML operator is best suited for measuring the focus quality of textured images.

7 Sampling the Focus Measure Function

The focus measure function of an image point (x, y) may be represented as $F(x,y,d)$. Since depth estimation is a local operation, we focus our attention on a single image point, bearing in mind that the same estimation method can be applied to all other image points. The focus measure function at the image point is $F(d)$. From the previous sections we know that $F(d)$ has a Gaussian distribution near the peak, with mean value $\overline{d}$ and standard deviation σ_F (Fig.10). The mean $\overline{d}$ corresponds to the stage displacement at which $F(d)$ is maximum, i.e. $F(\overline{d}) = F_p$. As the texture content on the surface element increases, F_p increases and σ_F decreases. Each surface element, therefore, is expected to have its own F_p and σ_F values.

If we use very small stage displacements ($\Delta d \approx 0$), the number of images to be obtained and processed is too large from the perspective of practical implementation. Hence, we use large displacements to obtain a few images of different focus levels and use the Gaussian model to inter-

polate a small number of focus measures in the peak region to obtain depth estimates. Computing focus measures at a finite number of stage displacements is equivalent to *sampling* the function $F(d)$; at each displacement d_i the focus measure $F(d_i)$ is computed to obtain the set $\{F(d_i) \mid i = 1, 2, ...M\}$. We show in the following section that a minimum of *three* focus measures are needed perform the Gaussian interpolation. Since the Gaussian model is valid only in the peak region of $F(d)$, these three focus measures must be computed in this region. This is ensured by using the condition $\sigma_F \leq \Delta d \leq 2\sigma_F$. Note that all object points are subjected to the same displacements. Therefore, by applying the above condition to the image area that has maximum texture content, we can ensure that a few or many focus measures will be computed in the $\pm\ \sigma_F$ range at all other image points.

The value of σ_F also increases with the depth of field of the imaging system. Therefore, for objects of larger dimensions also, only a small number of images may used by increasing the depth of field.

8 Depth Estimates from Focus Measures

We now describe the estimation of depth of a surface point (x,y) from the focus measure set $\{F(d_i) \mid i = 1, 2, ...M\}$. The parameter $\overline{d}$ represents the depth of the surface point. For convenience the notation F_i is used to represent the focus measure value $F(d_i)$. We present algorithms for two different depth estimation methods. Each algorithm may be applied to all points in the image to obtain depth maps.

8.1 Coarse Resolution Depth Estimation

The first algorithm simply looks for the displacement value d_i that maximizes the focus measure and assigns that value to $\overline{d}$.

Algorithm 1

Step 1: Let $k = 1$, $F_{max} = 0$.

Step 2: If $F_k > F_{max}$, then $F_{max} = F_k$ and $\overline{d} = d_k$.

Step 3: If $k < M$, then $k = k + 1$, go to step 2. Else stop.

Step 4: If $F_{max} < T_2$, the point (x,y) belongs to the background.

This simple algorithm may be used to compute low resolution depth estimates. The performance of the algorithm is directly dependent on the selection of Δd. If Δd is small, a large number of object images are obtained and the depth maps are more accurate.

8.2 Depth Estimation by Gaussian Interpolation

The second algorithm uses the Gaussian distribution to model the peak region of the focus measure function $F(d)$ and interpolates the computed measure values to obtain more accurate depth estimates. The following algorithm uses only three focus measures, namely, F_{m-1}, F_m, and F_{m+1}, that lie on the largest mode[6] of $F(d)$, such that, $F_m > F_{m-1}$ and $F_m > F_{m+1}$ (Fig.10).

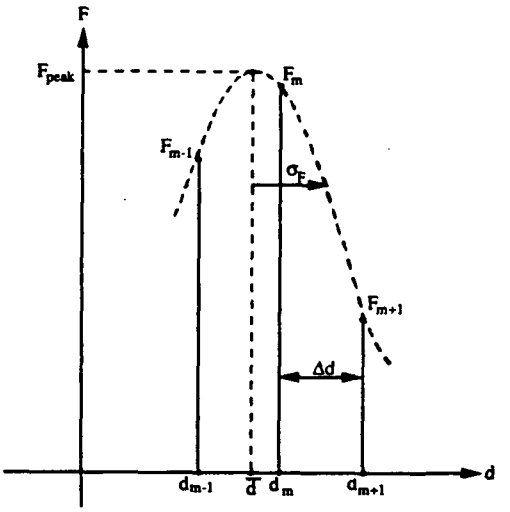


Figure 10: Gaussian interpolation of focus measures.

Using the Gaussian model, the focus measure function may be expressed as:

$$F = F_p\ exp\left\{-\frac{1}{2}\left(\frac{d-\overline{d}}{\sigma_F}\right)^2\right\} \tag{15}$$

where $\overline{d}$ and σ_F are the mean and standard deviation of the Gaussian distribution (Fig.10). Using natural logarithm, we can rewrite eq. 15 as:

$$ln\,F = ln\,F_p - \frac{1}{2}\left(\frac{d-\overline{d}}{\sigma_F}\right)^2 \tag{16}$$

By substituting each of the three measures F_{m-1}, F_m, and F_{m+1}, and its corresponding displacement value in eq. 16, we obtain three equations that can be solved for $\overline{d}$ and σ_F:

$$\overline{d} = \frac{(ln\,F_m - ln\,F_{m+1})(d_m{}^2 - d_{m-1}{}^2)}{2\,\Delta d\,\{(ln\,F_m - ln\,F_{m-1}) + (ln\,F_m - ln\,F_{m+1})\}} - \frac{(ln\,F_m - ln\,F_{m-1})(d_m{}^2 - d_{m+1}{}^2)}{2\,\Delta d\,\{(ln\,F_m - ln\,F_{m-1}) + (ln\,F_m - ln\,F_{m+1})\}} \tag{17}$$

[6]Due to image noise and variations in magnification, the focus measure function may be multi-modal with one strong peak and one or more weak ones.

$$\sigma_F{}^2 = -\frac{(d_m{}^2 - d_{m-1}{}^2) + (d_m{}^2 - d_{m+1}{}^2)}{2\{(\ln F_m - \ln F_{m-1}) + (\ln F_m - \ln F_{m+1})\}} \quad (18)$$

Using eq.15, we can find F_p from σ_F and $\overline{d}$ as:

$$F_p = F_m / exp\left\{-\frac{1}{2}\left(\frac{d_m - \overline{d}}{\sigma_F}\right)^2\right\} \quad (19)$$

If F_p is large and σ_F is small, the focus measure function has a strong peak, indicating high surface texture content in the vicinity of the image point (x,y). Hence, the values of F_p and σ_F can be used to segment the observed scene into regions of different texture content. Fig. 11 show the results of Gaussian interpolation applied to the focus measures computed for a real sample.

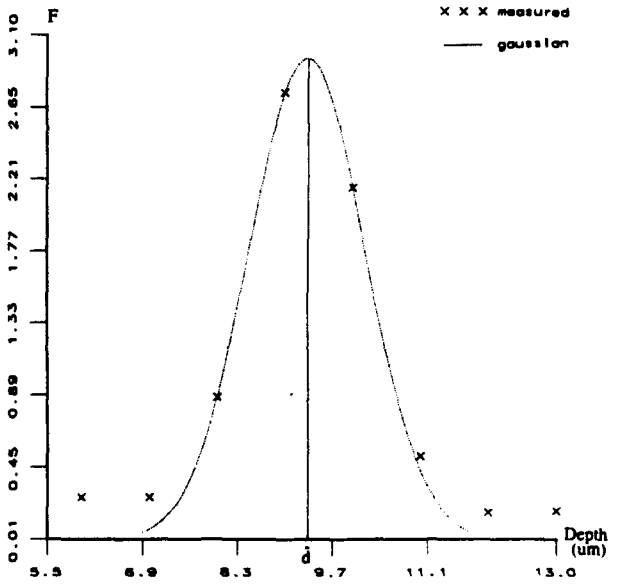


Figure 11: Gaussian interpolation: Experimental result.

The following algorithm first finds the measures F_{m-1}, F_m, and F_{m+1} that correspond to the strongest peak of $F(d)$, and then uses these measures to estimate the depth $\overline{d}$ by Gaussian interpolation.

Algorithm 2

Step 1: Let $k = 3$, $F_{m-1} = 0$, $F_m = 0$, $F_{m+1} = 0$, $d_m = 0$.

Step 2: If $F_{k-1} > F_m$, $F_{k-1} > F_k$, and $F_{k-1} > F_{k-2}$, then:

$F_m = F_{k-1}$
$F_{m-1} = F_{k-2}$
$F_{m+1} = F_k$
$d_m = d_{k-1}$

Step 3: If $k < M$, $k = k + 1$, go to step 2.

Step 4: $d_{m-1} = d_m - \Delta d$ and $d_{m+1} = d_m + \Delta d$. Determine $\overline{d}$, σ_F, and F_p using Eqs. 17, 18, and 19.

Step 5: If $F_p < T_3$ or $\sigma_F > T_4$, the image point (x,y) belongs to background. Stop.

Since the values of F_p and σ_F are only useful for texture segmentation, their evaluation may be avoided to save computations.

9 An Automated System

9.1 Implementation

We have implemented a fully automated shape from focus system for the recovery of microscopic objects. A photograph of the system is shown in Fig.12. A Nikon Alphaphot-2 model microscope is used to image the objects. Objects can be magnified using objective lenses with ×10, ×40, and ×100 magnification. The object is illuminated using bright field illumination where light energy is focused on the object by the same lenses that are used to magnify the object. A CCD camera with 512×512 pixels is mounted on the microscope to obtain digital images of the object. The z-axis of the microscope stage is driven by a stepper motor and the position of the stage can be computer controlled with a resolution and accuracy of 0.02 μm. The shape from focus algorithm is programmed and executed on a Sun SPARC 2 workstation.

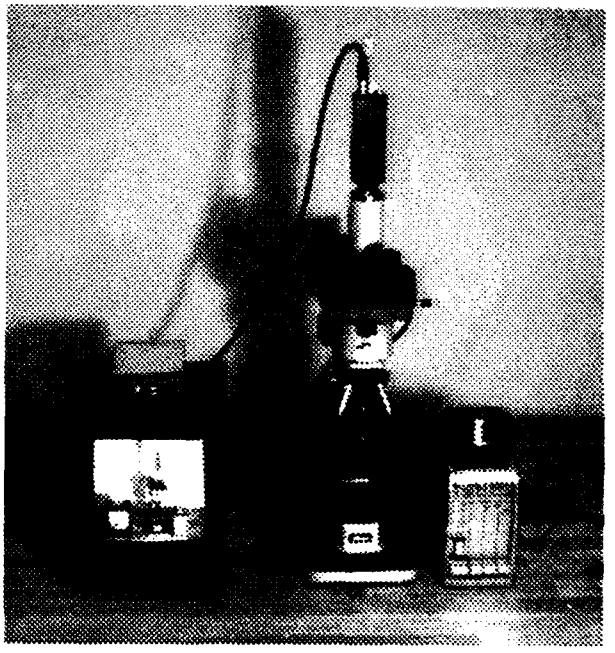


Figure 12: Automated shape from focus system.

The object is placed on the microscope stage and the appropriate objective lens is used to magnify the object.

The f ire parameters (T_1 and *step*) and the stage displ. Δd) are provided to the program. The program . .utomatically increments the stage position, digitizes and stores an image for each new position, and uses the image sequence to compute a depth map of the object. The program also reconstructs a focused image of the object from the sequence of defocused images. The reconstruction algorithm uses the estimated depth to locate and patch together the best focused image areas in image sequence.

9.2 Results

Prior to automating the shape from focus system, experiments were conducted to determine the accuracy and feasibility of the method. The first experiment was conducted on a steel ball sample that is 1590μm in diameter. The ball has a rough surface that gives it a textured appearance. A camera image of the ball under bright field illumination is shown in Fig.13(a). Incremental displacements of Δd = 100 μm were used to obtain 12 images of the ball, and a 5×5 SML operator was applied to the image sequence to obtain focus measures. Depth maps of the ball, generated by the coarse resolution and Gaussian interpolation algorithms, are shown in Fig.13(b) and 13(c), respectively. The known size and location of the ball were used to obtain error maps by subtracting a smooth ball from the two depth maps. It is difficult to define the accuracy of the method as it depends on several factors: the surface texture, depth of field of the imaging system, and the incremental displacement Δd. The table shown in Fig.13(d) shows the error statistics computed from the error maps corresponding to the two algorithms. A total of 23235 image pixels lie within the boundary of the ball. The number of depth values computed by each algorithm depends on the values selected for the thresholds T_2, T_3, and T_4.

The automated system has been used to recover the shapes of a variety of industrial samples. Fig.14 shows a tungsten paste filling in a via-hole on a ceramic substrate [16]. The filling is used to establish electrical connections between different components on a circuit board. Conditions such as excess filling and lack of filling cause electrical defects such as short and open circuits. The via-hole shown in Fig.14 is approximately 70 μm in diameter an is not sufficiently filled with tungsten paste. A total of 18 images of the via-hole were obtained using stage position increments of 4μm. Some of these images are shown in Fig.14(a-f). The specular reflectance properties and variable size of the tungsten particles gives the surface a random texture. The white background is the substrate area that has weak texture. A depth map of the via-hole was obtained using the Gaussian interpolation algorithm. A 5 × 5 median filter was used to remove a few erroneous depth estimates that resulted from the lack of texture in some image areas. Fig.14(g) and Fig.14(h) show the reconstructed image and two views of the depth map.

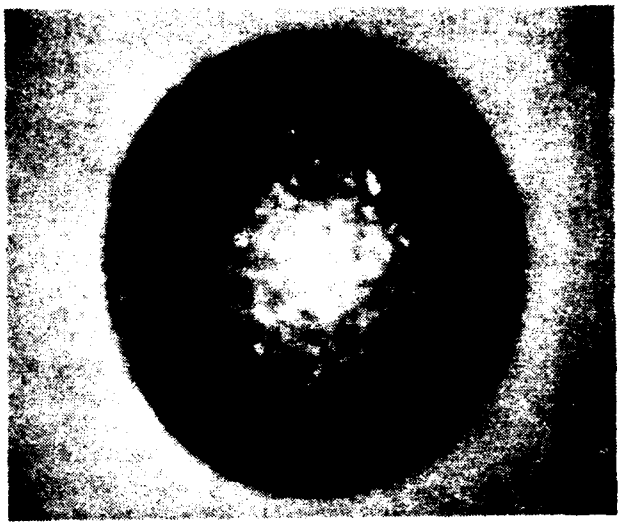
The object in Fig.15 is a contamination particle on the surface of a circuit board. The particle if approximately 250 μm in length and 150 μm in height. Though the particle surface produces images with very weak texture, the reconstructed image and the depth map generated by the system are accurate and detailed.

The above experiments indicate that the Gaussian interpolation algorithm performs stably over a wide range of textures. Errors in computed depth estimates result from factors such as image noise, Gaussian approximation of the SML focus measure function, and weak textures in some image areas. We are currently using the system to recover the shapes and focused images of a variety of microscopic biological samples, including micro-organisms and chromosomes.

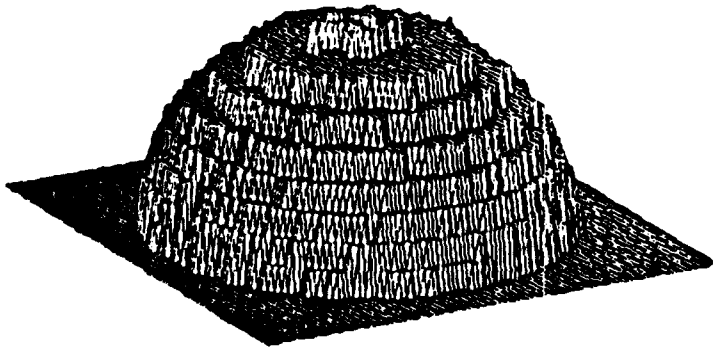
10 Summary

We conclude with a brief summary of the shape from focus method presented in this paper.

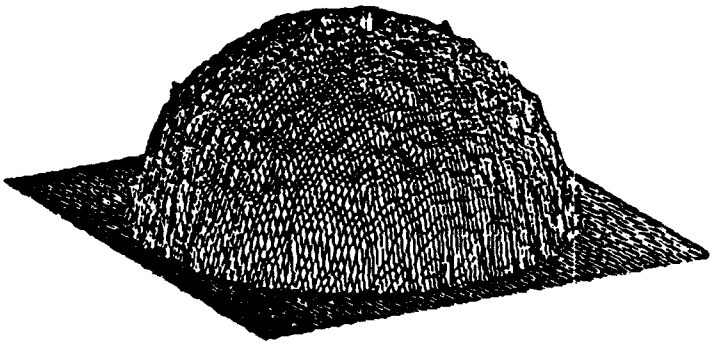
- Most surfaces appear rough at some level of magnification. Visibly rough surfaces produce highly textured images and it is difficult to recover the three-dimensional shapes of these surfaces using existing passive and active sensing techniques.
- The shape from focus method obtains a sequence of object images by translating the object through the focused plane of the imaging system. The sum-modified-Laplacian operator is applied to the image sequence to compute a set of focus measures at each point on the object surface.
- Using a model for the focus measure function, the focus measures at each point are interpolated to compute accurate depth estimates. The local nature of the depth estimation technique enables it to adapt to substantial variations in image texture.
- A fully automated shape from focus system has been developed. The system has been used to recover dense and accurate depth maps of several microscopic objects.



(a) Camera image.



(b) Depth map: coarse resolution.

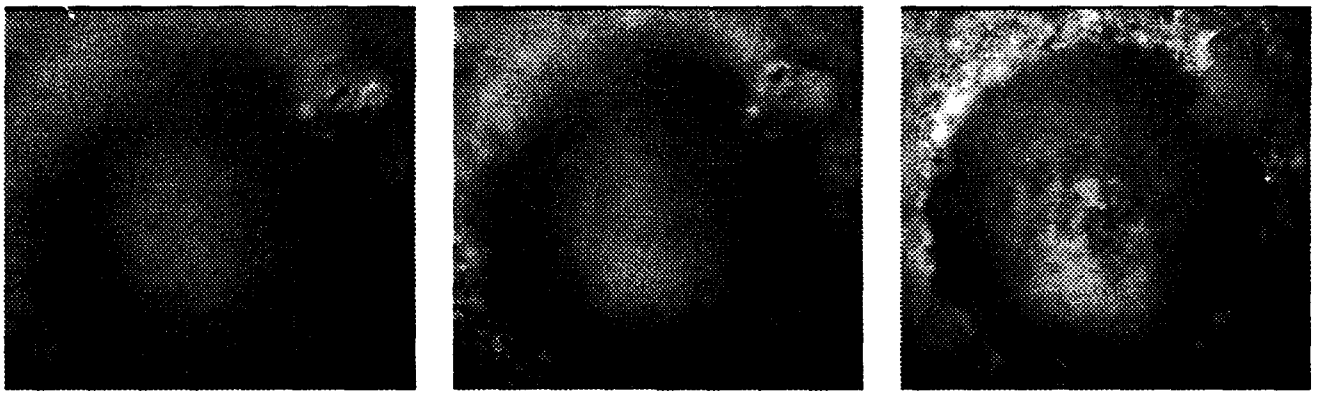


(c) Depth map: Gaussian interpolation.

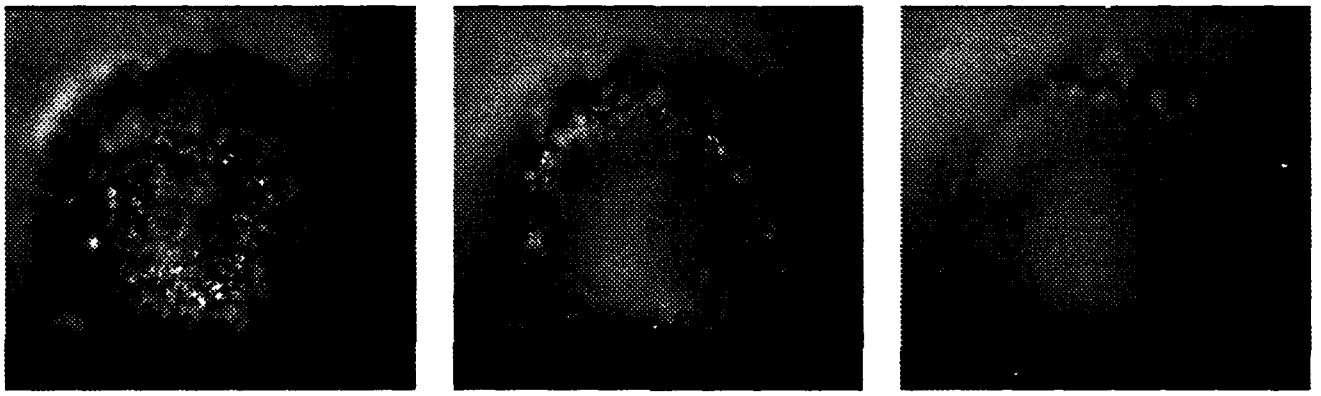
Diameter of Test Sphere : 1590 μm		
	Coarse	Gaussian Interpolation
Number of Points	22682	23257
Mean Error (μm)	7.861	3.857
Mean Absolute Error (μm)	30.32	13.815
Maximum Absolute Error (μm)	187.80	175.82

(d) Error statistics.

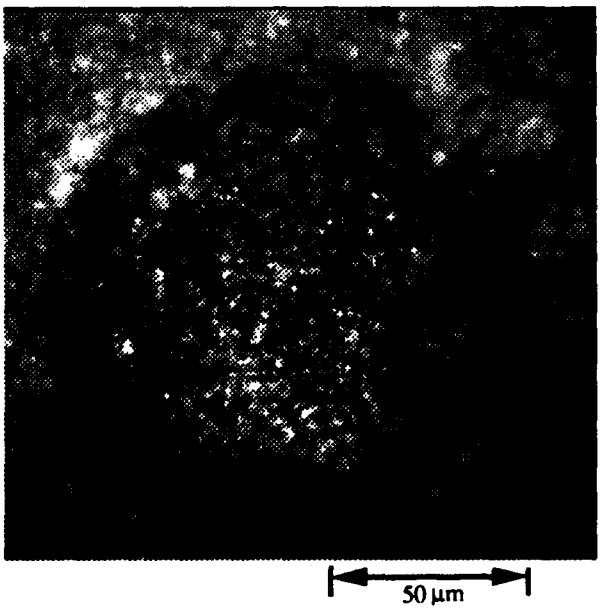
Figure 13: Results: Steel ball.



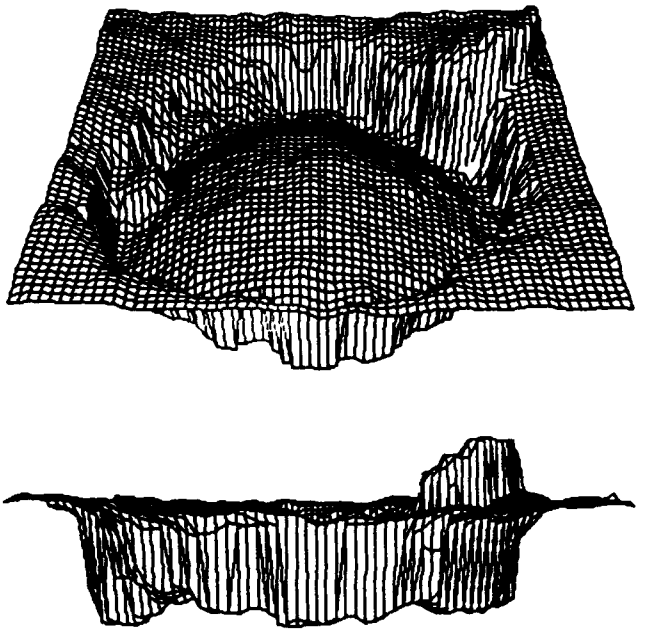
(a) i=2 (b) i=5 (c) i=8



(d) i=11 (e) i=14 (f) i=18

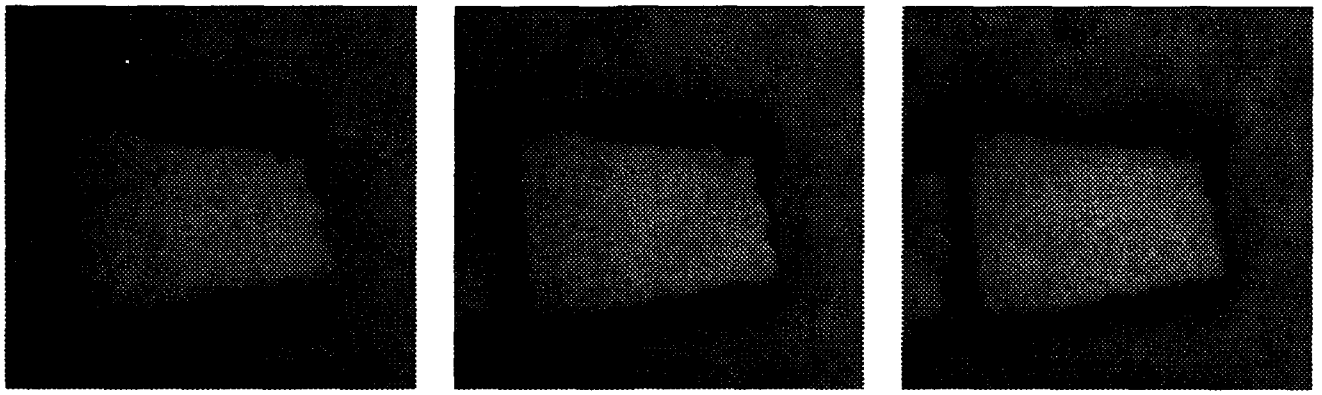


(g) Reconstructed Image.



(h) Depth maps.

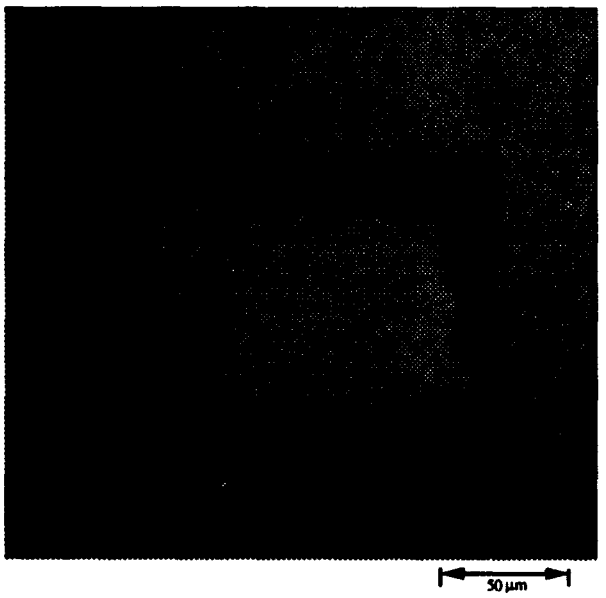
Figure 14: Results: Via-hole filling.



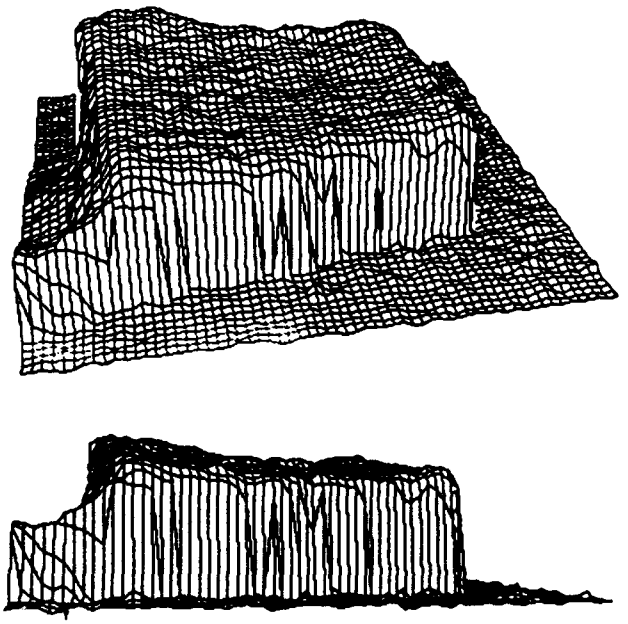
(a) i=2 (b) i=5 (c) i=8



(d) i=11 (e) i=14 (f) i=18



(g) Reconstructed Image.



(h) Depth maps.

Figure 15: Results: Contamination particle on a circuit board.

Acknowledgements

This research began during the author's visit to the Production Engineering Research Lab. (PERL), Hitachi Ltd., Yokohama, Japan. The author is grateful to Dr. Yasuo Nakagawa for his support and encouragement. The author thanks Ushir Shah for his contribution to the implementation of the automated system. This work has benefited from several discussions with Reg Willson of the VASC Group at CMU.

References

[1] B.K.P. Horn, *Focusing,* MIT Artificial Intelligence Laboratory, Memo No. 160, May, 1968.

[2] J.M. Tenenbaum, *Accommodation in Computer Vision,* Ph.D. Thesis, Stanford University, 1970.

[3] R.A. Jarvis, *Focus optimization criteria for computer image processing,* Microscope, Vol. 24, No. 2, pp. 163-180, 1976.

[4] J.F. Schlag, A.C. Sanderson, C.P. Neumann, F.C. Wimberly, *Implementation of automatic focusing algorithms for a computer vision system with camera control,* Carnegie Mellon University, CMU-RI-TR-83-14, August, 1983.

[5] E. Krotkov, *Focusing,* International Journal of Computer Vision, Vol. 1, pp. 223-237, 1987.

[6] E. Krotkov, *Exploratory visual sensing with an agile camera,* Ph.D. Dissertation, TR-87-29, University of Pennsylvania, 1987.

[7] A. Pentland, *A New Sense for Depth of Field,* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 9, No. 4, pp. 523-531, July 1987.

[8] P. Grossmann, *Depth from Focus,* Pattern Recognition Letters, Vol. 5, pp. 63-69, 1987.

[9] T. Darrell and K. Wohn, *Pyramid Based Depth from Focus,* Proc. CVPR, pp. 504-509, 1988.

[10] M. Subbarao, *Efficient Depth Recovery through Inverse Optics,* Machine Vision for Inspection and Measurement, edited by H. Freeman, Academic Press, 1989.

[11] T. Ohta, K. Sugihara, and N. Sugie, *A Method for Image Composition using Image Variance,* Transactions of IECE, Vol. J66-D, No. 10, pp. 1245-1246.

[12] K. Kaneda, Y. Wakasu, E. Nakamae, E. Tazawa, *A Method of Pan-Focused and Stereoscopic Display Using a Series of Optical Microscopic Images,* Proc. of Fourth Symposium on Image Sensing Technologies in Industry, pp. 189-194, June, 1988.

[13] K. Torrance and E. Sparrow, *Theory for Off-Specular Reflection from Roughened Surfaces,* Journal of the Optical Society of America, No. 57, pp. 1105-1114, 1967.

[14] S. K. Nayar, K. Ikeuchi, T. Kanade, "Surface Reflection: Physical and Geometrical Perspectives," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* pp. 611-634, July, 1991.

[15] S. K. Nayar, *Shape from Focus,* Carnegie Mellon University, CMU-RI-TR-89-27, November, 1989.

[16] T. Ninomiya, M. Nomoto, Y. Nakagawa, *Automatic 2-1/2D Shape Inspection System for Via-Hole Fillings of Green Sheets by Shadow Image Analysis,* Proc. IEEE Intl. Conf. on Robotics and Automation, pp. 515-520, 1989.

[17] M. Born and E. Wolf, *Principles of Optics,* London:Pergamon, 1965.

[18] B. K. P. Horn, *Robot Vision,* MIT Press, 1986.

[19] G. Westheimer, *The Eye,* Medical Physiology, Thirteenth Edition, Mosby Co., 1974.

[20] R. G. Willson and S. A. Shafer, *Dynamic Lens Compensation for Active Color Imaging and Constant Magnification Focusing,* Carnegie Mellon University, CMU-RI-TR-91, October, 1991.

Hierarchical Decomposition and Axial Representation of Shape*

Hillel Rom and Gérard Medioni
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273
Email: rom@iris.usc.edu

Abstract

We address the problem of obtaining natural (intuitive) descriptions of planar shapes. Shape description is a major problem in machine perception, and is the basis for recognition. Many approaches have been suggested, but none provide a complete and natural solution. In this paper we suggest a method for producing an axial representation of a shape based on a hierarchical decomposition of the shape into its parts. The novelty of our approach lies in the combination of several competing approaches and tools, into a unified scheme and an efficient implementation producing natural descriptions. We use Smooth Local Symmetries for the axial representation of parts. We also use parallel symmetries to provide information on global relationships within the shape. This information is used for parsing the shape into a hierarchy of parts. Currently we assume that our shape is a closed planar curve. Our approach uses both region and contour information, can handle shapes with corners, and addresses the issues of local vs. global information, the issue of scale and the notion of part. Our method is computationally efficient, parameter free, stable, and we present results which show that it provides an intuitive shape description.

1 Introduction

Shape description is the basis for recognition, and is one of the key problems in machine perception. Various methods for shape description have been suggested through the years of research in machine and human perception, but none provide a complete and natural solution to the problem.

Many researchers have discussed the requirements from a good shape description. Examples can be found in [2, 14, 15, 13, 8, 17]. A good description should be rich, stable and invariant to changes in the viewing conditions. In addition, it should be capable of describing partially occluded parts. These requirements lead to descriptions which are segmented and hierarchical.

*This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-90-C-0078. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

Hoffman and Richards [8, 18], suggest a contour based representation. Following psychophysical observations, they segment the contour into segments, called *codons*, at negative curvature minima. Curvature maxima and inflections are used for internal part description. This scheme, being contour based, does not incorporate any region information. It also does not account for any global relationships between distant sections of the curve.

Many authors have suggested a representation scheme based on a skeleton or an axis around which the shape is locally symmetric. Examples are: Blum's *Symmetry Axis Transform (SAT)* [3], Brooks's *Generalized Ribbons* [5], which are a two-dimensional version of Binford's *Generalized Cylinders* [2], and Brady's *Smooth Local Symmetries (SLS)* [4]. These methods incorporate contour and region information. An in depth comparison of these three methods was performed by Rosenfeld [20], and extended by Ponce [16]. These methods are local in nature, they are very sensitive to noise, and they are expensive to compute, even when using analytical approximations to the shape.

Leyton [12] has suggested a process-grammar for shape. He argues that a shape can be understood as the outcome of processes that formed it. After proving an important duality between curvature extrema and symmetry structure, he uses symmetry axes to infer process history. This approach assumes all shapes are basically circles which have been deformed by protrusions and indentations. No distinction is made between parts and protrusions, and global relationships within the shape are ignored, making this description unnatural in many cases.

Recently, Kimia *et al.* [10] have proposed a theory of shape based on a reaction-diffusion equation. Their method produces a hierarchical decomposition of shape into parts and protrusions. The issues of part and protrusion, as well as the issue of scale are addressed. However, it is a continuous process which is computationally expensive. In addition their scheme reduces every part to a circle. It provides only a parsing of the shape into parts not a description of the parts.

We are interested in producing *natural* descriptions of planar shapes. Our ultimate goal is in achieving human like performance. In this paper we suggest a method for producing an axial representation of a shape, along

with a hierarchical decomposition of the shape into its parts. The novelty of our approach lies in the combination of several, often competing approaches and tools, into a *unified scheme* and an *implementation* producing natural descriptions. We use SLS for the axial description of parts. We also use *parallel symmetries* [23] to provide information on global relationships within the shape. This information is used to parse the shape into a hierarchy of parts. When conflicts between the local interpretation and the global interpretation arise, we generate descriptions based on both interpretations. Currently we assume that our shape is a closed planar curve which we approximate using quadratic B-splines. Our approach is region and contour based, and addresses the issues of local and global information, the issue of scale and the notion of part. Our method is computationally efficient, parameter-free, stable, and we present results which show that it provides intuitive shape descriptions.

The next section outlines the basic ideas of our approach. Section 3 presents the details of our approach and implementation. Examples of the results we obtained are presented in Section 4 followed by some concluding remarks in Section 5.

2 Overview of the Approach

We suggest a hierarchical approach for shape description, combining local and global information. We produce a decomposition of the shape into parts together with an axial description of these parts. We outline our method here, the details of our implementation are presented in the next section.

A strategy often used for obtaining axial representations, is to generate all possible axes, then to select the appropriate ones according to some criteria [6, 17]. Some axes, unfortunately, cannot be found until some parts are removed. It is necessary to use a hierarchical strategy instead, in which, at each step, local and well defined parts are described and removed. Once these parts are removed, the next level parts can now be described. This process is efficient and produces a decomposition of the shape into its intuitive parts with a stable axial description of these parts. One remaining problem with this approach is that it ignores global relationships between different parts of the shape. We use *parallel symmetries*, introduced by Ulupinar and Nevatia [23], to detect such global relationships. Once conflicts between the local description and the global relationships are detected, we create a branch in our decomposition process, thus allowing multiple plausible interpretations, which turn out to be rare.

Figure 1 presents an outline of the data flow of our approach. Initially, we perform some preprocessing on the image including edge detection and linking. We then approximate the input shape with approximating B-splines. Since we are interested in an axial representation, we find SLS to be a powerful descriptive tool. However, applied globally on the entire shape it produces noisy and poor descriptions. Therefore, we first segment the contour at curvature sign changes into initial local parts. We represent these parts using SLS. A theorem proved by Leyton [11], relates the existence and

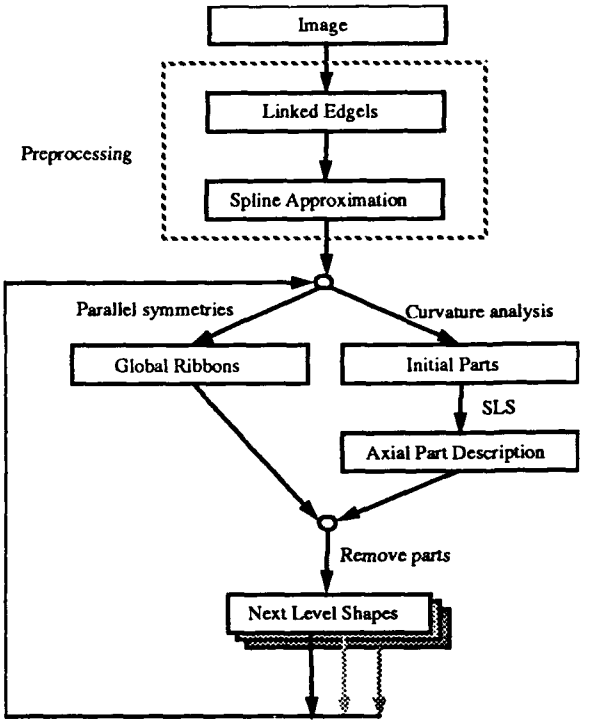


Figure 1: Outline of the data flow in the system.

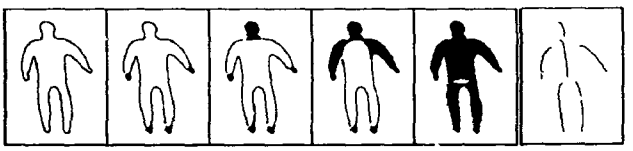


Figure 2: The decomposition of a man shape. The original shape is on the left. At every step the parts already explained are shaded. The skeleton description is also presented.

uniqueness of an SLS axis describing these parts to the curvature extrema of their contour. This enables us to efficiently generate stable and noise free descriptions for these local parts. In addition to this local description, we find global relationships within the shape, by computing the parallel symmetries.

Now that we have the local parts and the information on the global relationships, we decompose the shape into parts. The decomposition is done hierarchically, first removing the small and well defined parts, and then analyzing the remaining shape. We generate the possible parsings of the shape into parts, starting from the initial shape and producing the axial representation determined by every parsing.

In order to clarify what we mean by an hierarchical decomposition, we present an example here (we present it in more detatil in Section 4). Figure 2 presents the decomposition of a *man* shape. The original shape is on the left. At every step the parts which were previously removed (explained) are shaded. The current shape not

yet explained is left white. The skeleton description obtained is also presented.

3 Issues and Implementation

In this section we describe the details of our approach and implementation. We first present the tools we use and the motivation behind our choice of these tools, then how these tools are combined into the shape decomposition process producing the hierarchical shape description.

3.1 B-Spline Approximation

Following Saint-Marc and Medioni [21, 22], we use approximating B-splines to represent planar curves. As noted there, this representation is rich, compact, stable and local. In the rest of this paper, we ignore the problem of obtaining this B-spline representation. From now on, we assume that our input consists of a *single closed* planar curve, represented by B-spline segments.

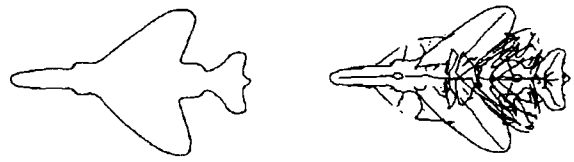
A B-spline is a piecewise polynomial which is expressed as a linear combination of polynomial basis functions. The coefficients, or *control points*, are the vertices of the B-spline *guiding polygon*. In our application we use quadratic B-splines, every curve segment is a quadratic polynomial depending on three adjacent control points.

Besides providing a simple analytical representation, the following features make this approximation attractive for our application:

- The B-spline is easily manipulated by modifying its guiding polygon.
- The B-spline is defined locally, changing the position of a vertex of the guiding polygon does not have a global effect on the representation.
- Quadratic B-splines are continuous and smooth. Discontinuities of the tangent could be introduced by using multiple control points.
- Each quadratic spline has *constant sign* curvature. This implies that zero crossings of curvature can occur only at the *knots* (the connection points between two splines).
- Quadratic splines have either one or no curvature extrema. As shown later, limits on the number and location of the curvature extrema simplify the complexity of our process significantly.

3.2 Smooth Local Symmetries

Brady and Asada [4] have introduced *Smooth Local Symmetries (SLS)* as a method for shape representation. Two points on a planar curve form a *local symmetry* if the line between them produces equal angles with the normals to the curve at the two points respectively. The line between the two points, is known as the *cross section*. The loci of the mid points of the cross sections, are the *symmetry axes*. A symmetry axis together with its cross sections is sometimes called an *SLS ribbon* or a *Brady ribbon*. We say that the region of the shape covered by the ribbon is *explained* by the ribbon.



(a) Original shape. (b) All SLS axes.

Figure 3: Computing all SLS is expensive and does not result in a good description.

Two of the major problems with using SLS are sensitivity to noise and the computational difficulty of recovering the SLS axis. Even in the case of analytical curves, it is generally impossible to find analytical solutions for the symmetry axis. Brady and Asada [4] have overcome these problems by approximating the curve with circular arcs. Circular arcs, however, cannot guarantee the C^1 continuity of the approximation.

Our curve representation is based on quadratic B-splines which are C^1 continuous. Since we are unable to solve analytically for the SLS of two quadratic curve segments (this requires finding the roots of a polynomial of degree six), we first find the axis termination points using search, and then interpolate for the rest of the axis. The complexity of the algorithm for computing the SLS of the whole shape is $O(n^2)$, where n is the number of quadratic curve segments in the spline approximation. This is similar to the circular arc case, but with a larger constant. For a complete description of the algorithm, refer to [19].

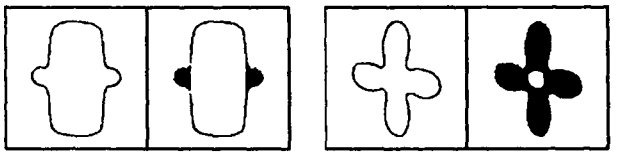
3.3 Parts

We believe that segmentation into parts is the key to any shape description used by higher tasks, such as recognition. However, a well known problem is that although a good description may help segmentation, the segmentation may be needed for obtaining a good description.

Computing the SLS within the whole shape results in many axes which are ambiguous or not relevant for a natural representation. For example, two very distant segments may also create a symmetry axis, which may even lie outside the shape. Figure 3 shows an example of an airplane shape and all of its SLS axes. Grouping the axes and selecting the significant ones is a difficult and computationally expensive problem (for an example of such an implementation refer to [6]). Therefore, we suggest that the shape be segmented into "natural" parts prior to the computation of the SLS. The SLS are then computed within the parts only. This, of course, forces us to address the issue of defining parts.

The Definition of a Part

Hoffman and Richards [9], and more recently Biederman [1] argue that a segmentation into parts occurs at the negative minima of curvature. This corresponds to the human intuition about parts, and is based on the fact, known as *transversality* [7], by which, in general, concavities arise when two convex volumes are arbitrar-



(a) Two Parts and a blob. (b) Four parts.

Figure 4: Segmenting at curvature minima results in 4 parts in both cases.

ily joined. Kimia *et al.* [10], however, note that although parts are bounded by curvature minima, the converse does not necessarily hold. See for example Figure 4: We perceive the two shapes to have different structure, one being composed of two parts and a large blob, and the other is composed of four similar parts. However, segmenting at curvature minima results in four parts in both cases.

We also recognize the importance of the transversality principle for the perception of parts. A positive curvature curve section bounded by negative curvature curve sections would imply a possible part in the process of interpreting the shape. We overcome the above problem raised by Kimia *et al.* by using a hierarchical decomposition, in which the smaller parts are removed before the larger ones are analyzed. Note that in contrast to Hoffman and Richards, we believe that segmenting exactly at negative curvature minima does not always correspond to the intuition of the parts termination. The final delineation of each part (the exact termination points), is based on the symmetry axis describing it.

Describing Parts

A major drawback of the descriptive power of the SLS is that, in general, the local symetries are not unique. This may lead to several ribbons, giving different explanations to the same regions of the shape. The *Symmetry-Curvature Duality Theorem*, proposed and proved by Leyton [11], relates the curvature of the curve and the axis of symmetry describing its shape. The theorem guarantees the existence and uniqueness of an SLS axis describing parts with one positive curvature extrema. For any additional positive curvature extremum within the part, there exists an additional axis going into it.

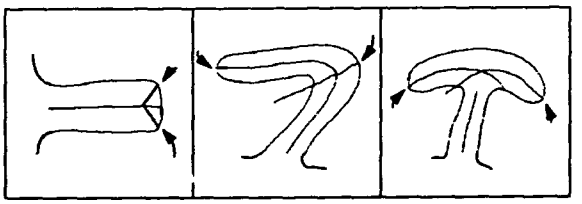
Given a shape, we first segment the contour into sections bounded by consecutive negative curvature curve sections. These sections are considered parts. We compute the axes of symmetry of all parts. From Leyton's theorem, we now know the origins of the axes, and we also know on which side of the ribbon every quadratic segment may be. Therefore, we have very few comparisons to make and we can recover the SLS axes explaining each part very efficiently. Figure 5 shows an example of a shape, and its initial segmentation into parts with their axial description.

Special Cases: Terminations, Bends, T-shapes

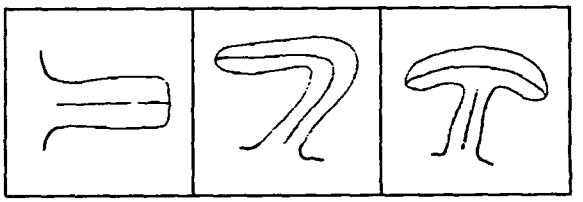
Parts which have more than one positive curvature extremum within them are initially described with more than one SLS axis. We have identified three special cases, which occur frequently, where a more intuitive single axis



Figure 5: A shape and the axial representation of its parts.



(a) SLS axes (The arrows point to positive curvature maxima)



(b) Selected axis for description

Figure 6: Multiple extrema: Termination, Bend, T-shape

can be found to describe these parts. Refer to Figure 6 for examples of these cases which we have intuitively named: *Termination*, *Bend*, and *T (or Mushroom)*. Figure 6a presents the SLS axial description of the corresponding shapes. Figure 6b presents our preferred description.

Once a part with two maxima of curvature is detected the three possible interpretations are tested and the best is chosen as the parts description. In our current implementation the interpretations are compared in terms of the variance of their cross section function. The results in Figure 6b and in Section 4 demonstrate that the correct decision is made by using this simple criterion.

Removing Parts

To produce an hierarchical decomposition, we need a mechanism for removing parts from the shape (see Section 3.5). Given a part, it is not clear how one removes it "gracefully". The direct method of simply connecting the two minima does not always produce the intuitively correct result. By removing control points of the guiding polygon we are able to remove parts efficiently and locally. Figure 7 demonstrates the performance of the above removing procedure on a shape and on its guiding polygon (refer to [19] for more details).

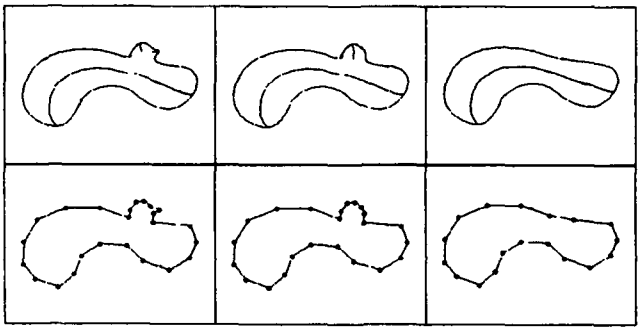


Figure 7: Removing parts: The shape and its guiding polygon.



(a) A snake shape. (b) Description using only local information. (c) Description using global relationships.

Figure 8: Global relationships are important for natural descriptions.

3.4 Representing Global Relationships

In Sections 3.2 and 3.3, we have presented SLS as a method for describing local parts. Leyton's theorem provides the basis for the efficient computation of the axes in these parts. Unfortunately, this description provides only local information. It is clear that this cannot suffice for an intuitive description. Consider for example the "snake" in Figure 8a: Using only the local information the snake would be described as a composition of 7 different parts (Figure 8b). We, however, would prefer to capture the global relationship and describe the snake as a single entity using the more intuitive symmetry axis, as in Figure 8c.

We have found *parallel symmetries*, suggested by Ulupinar and Nevatia [23], to be very useful in capturing global relationships within planar shapes. Let $C_i(s) = (x_i(s), y_i(s))$, for $i = 1, 2$ be two parametric planar curves, and let $\theta_i(s)$ be their tangent orientation. $C_1(s)$ and $C_2(s)$ are said to be *parallel symmetric* if there exists a continuous monotonic function $f(s)$ such that $\theta_1(s) = \theta_2(f(s))$. The symmetry axis is the loci of the mid points of the cross sections between $C_1(s)$ and $C_2(f(s))$ for all s where $f(s)$ exists. The symmetry axis together with the cross sections is a *parallel ribbon*. We say that the part of the shape covered by the ribbon is *explained* by the ribbon. Figure 8c is an example of a parallel symmetry axis, in this case the two curves are sections of the contour of the same object. As shown by Saint-Marc and Medioni [21, 22], the detection of parallel symmetries between quadratic B-splines is computationally very efficient. Using their algorithm, we compute all elementary parallel ribbons. From these elementary ribbons, we select the parallel ribbons which are significant for the description of the shape. We shall often refer to these significant parallel ribbons as *global ribbons*. The selection is done based on some simple and intuitive filtering and grouping criteria (see [19] for details).

Once such a global ribbon (significant parallel ribbon) is found, it is important to record its relationships with the local parts. A local ribbon (part) may continue the global ribbon, thus the two ribbons support each other in the parsing. A local ribbon may also conflict with the global ribbon, when both ribbons give a different explanation of the same region of the shape. In this case they cannot be used together to describe the shape.

3.5 Shape Decomposition

In this section we describe the decomposition of the shape, a planar closed curve, into a hierarchy of parts, based on the size of the parts and on global relationships.

The shape is decomposed using a recursive procedure: Given the current shape, we compute all local parts, as defined in Section 3.3, and all the global ribbons, as defined in Section 3.4. Every local part is represented by an SLS ribbon (axis and cross sections), and every global ribbon is represented by a parallel ribbon and pointers to its local continuations and local conflicts. We now create a new shape, generated from the current shape by removing its *smallest* parts in parallel. In case of a conflict with a global ribbon we generate the two possible interpretations, the first ignoring the global relationships, and the second considering the global relationships. (We do not attempt to decide which is the correct interpretation. This decision usually requires higher level knowledge.) Under the second interpretation a new shape is generated from the current shape by removing its smallest parts, this time using the information on the global relationships. The global information is used by assigning high size values to parts which are related to the global ribbon, either continuing or conflicting with the global axis. Therefore, these parts are removed only if there are *no* local parts which are unrelated to the global ribbon. Note that the second interpretation is generated only if the removed parts are different from the removed parts in the "all local" case. The number of interpretations is, therefore, bounded by $n + 1$ where n is the number of global ribbons in the shape. However, in most cases there are no significant global relationships within the shape, resulting in unique parsing of the shape. The process is continued recursively on each branch.

Since we only remove parts, the complexity of the shape, in terms of the number of control points of the guiding polygon, reduces from level to level, and the process is bound to terminate rapidly. The recursive procedure is stopped at one of the following atomic cases: The shape is empty (in the previous step all parts had about the same size), the shape is a positive curvature blob (no zero crossings of curvature), or the shape has exactly two zero crossings of curvature (a bean shape).

The axes of the parts removed are accumulated down the decomposition hierarchy. Therefore, at every atomic level we have a skeleton representation of the shape, given by the single parsing of the shape, defined by the path from the original shape to that atomic state.

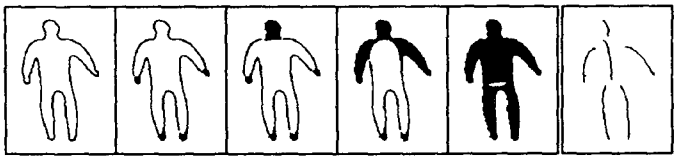


Figure 9: The decomposition of a man shape. The original shape is on the left. The skeleton description is on the right (see text).

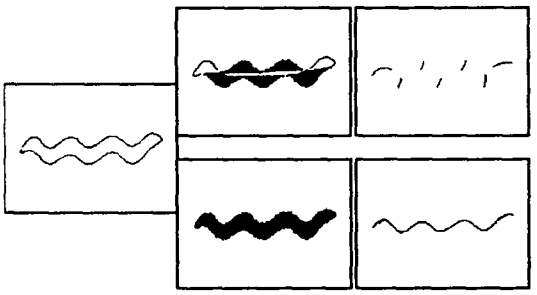


Figure 10: The decomposition of a snake shape. The original shape on the left. The two possible skeleton descriptions are on the right (see text).

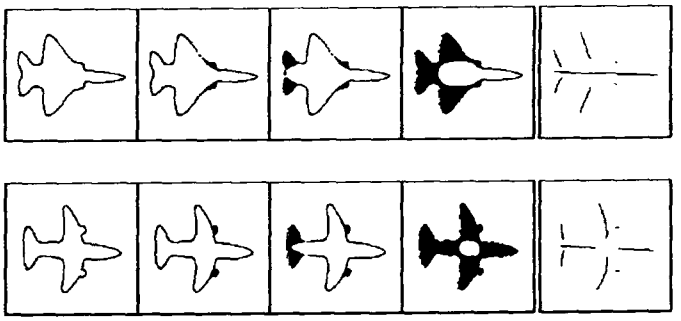


Figure 11: The decomposition of airplane shapes.

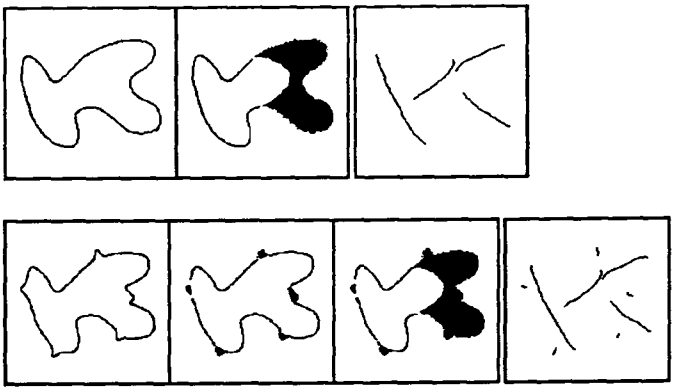


Figure 12: The stability of the process (see text)

4 Experimental Results

We have applied our method to several shapes. For each, we present the decomposition, which is the output of our description process. The original shape is on the left. At every node the parts which were previously removed are shaded. The current shape, not yet explained, is left white. As explained earlier, in Section 3.5, the axial representations of the parts removed are accumulated. Therefore, every interpretation generates a skeleton description of the shape. That skeleton description is presented at the end of the decomposition on the right.

The first example is the man shape (Figure 9). Since there are no significant global ribbons in this shape and in its sub shapes, there are no branches in the interpretation process. There is a unique decomposition of the shape into a hierarchy of parts. Note that, since in our current implementation we define the size of the part to be the length of the axis, at the last step the legs and the torso are removed in parallel, leaving the final blob at the last stage.

The next example is the snake shape (Figure 10). This is an example of the importance of global relationships. Simply segmenting at minima of curvature (e.g. Richards and Hoffman [18]), or interpreting the shape as a result of a history of protrusions and indentations (e.g. Leyton [12]), results in an unintuitive (but possible) description, as in the top interpretation path. Using the global ribbon, we are able to produce the intuitive description on the bottom path.

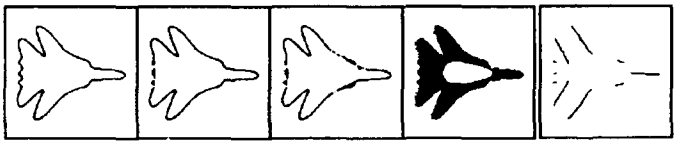
In the next two examples, the sketches of airplanes in Figure 11, there is once again no significant global information conflicting with the local one. Therefore, they each have a unique decomposition leading to the skeleton description.

The next example illustrates the inherent stability of our hierarchical approach. In Figure 12a we present the decomposition of an unspecified shape. We have taken this shape and added several bumps to its curve. A good shape description mechanism should be stable enough to produce a description which is similar to the original and yet rich enough to show the differences. As Figure 12b shows, our algorithm performs well on this example, by detecting and removing all the small bumps in the first step, after which the process is identical to the original case. A higher level task interpreting these results could easily examine the similarities and differences between the two shapes.

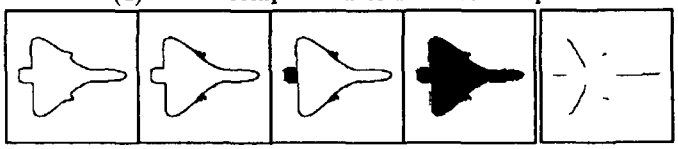
All the examples we have presented so far were based on synthetic data. We have also applied our method to shapes obtained from real images of model airplanes taken on a light table. The results are presentd in Figure 13. Note also that the *"F16"* shape of Figure 13d has sharp corners (i.e. discontinuities of the tangents). These corners are simply represented using multiple control points in the guiding polygon. They are considered as any other curvature extrema, and do not require any special treatment in the decomposition process.

5 Concluding Remarks

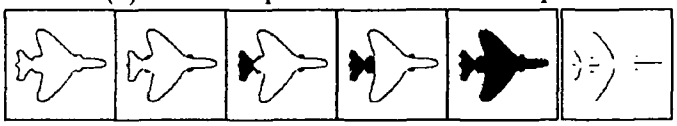
We have presented a powerful method for obtaining natural descriptions of planar shapes. Our method produces an axial representation of a shape, and a discrete hierar-



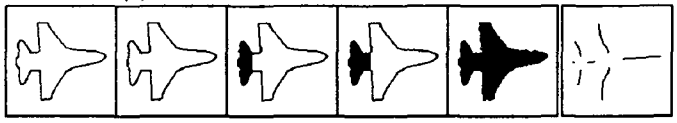
(a) The decomposition of an "F14" shape.



(b) The decomposition of an "F106" shape.



(c) The decomposition of an "F4" shape.



(d) The decomposition of an "F16" shape.

Figure 13: The decomposition of airplane shapes obtained from real data.

chical decomposition of the shape into its parts. We use Smooth Local Symmetries for the axial representation of parts. We also use parallel symmetries to provide information on global relationships within the shape. This information is used to parse the shape.

Our approach is both region and contour based, combines local and global information, can handle corners, and addresses the issue of scale and the notion of part. Our method is computationally efficient, parameter free and stable. We have presented results which show that it provides intuitive shape descriptions for various shapes.

Currently we assume that our shape is a closed curve. We do not solve the figure-ground problem. It is clear that for many applications this problem has to be solved prior to shape description. In addition, due to artifacts of the spline representation, our current implementation may not be stable on straight lines. This is because when approximating a straight line, the spline approximation may fluctuate between positive and negative curvature segments. This may be solved by ignoring such changes of the sign of the curvature if the absolute values are very small. We also consider only positive curvature parts. For intuitive descriptions it is sometimes needed to capture negative curvature parts. For example, our method will not work well on the famous example of a rectangle with a sharp indentation. These problems and others are subject to further research.

References

[1] I. Biederman. Recognition by components. *Psychological Review*, 94:115–147, 1987.

[2] T. O. Binford. Visual perception by computer. In *IEEE Conference on Systems and Controls*, Miami, FL, 1971.

[3] H. Blum. *A Transformation for Extracting New Descriptors of Shape.* MIT Press, Cambridge, MA, 1967.

[4] M. Brady and H. Asada. Smoothed local symmetries and their implementation. *The International Journal of Robotics Research*, 3(3):36–61, 1984.

[5] R. A. Brooks. Symbolic reasoning among 3-D models and 2-D images. *Artificial Intelligence*, 17:285–348, 1981.

[6] J. H. Connell and M. Brady. Generating and generalizing models of visual objects. *Artificial Intelligence*, 31(2):159–183, 1987.

[7] V. Guillemin and A. Pollack. *Differential Topology.* Prentice Hall, 1974.

[8] D. D. Hoffman. *Representing Shapes for Visual Recognition.* PhD thesis, Massachusetts Institute of Technology, May 1983.

[9] D. D. Hoffman and W. A. Richards. Parts of recognition. *Cognition*, 18:65–96, 1985.

[10] B. B. Kimia, A. Tannenbaum, and S. W. Zucker. Toward a computational theory of shape: An overview. In *Proceedings of European Conference on Computer Vision*, pages 402–407, Antibes, France, 1990.

[11] M. Leyton. Symmetry-Curvature duality. *Computer Vision, Graphics and Image Processing*, 38:327–341, 1987.

[12] M. Leyton. A proces-grammar for shape. *Artificial Intelligence*, 34:213–247, 1988.

[13] D. Marr. *Vision.* W. H. Freeman and Co., San-Fransisco, CA, 1982.

[14] D. Marr and K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London*, B(200):269–294, 1977.

[15] R. Nevatia. *Machine Perception.* Prentice Hall, 1982.

[16] J. Ponce. On characterizing ribbons and finding skewed symmetries. *Computer Vision, Graphics and Image Processing*, 52:328–340, 1990.

[17] K. Rao. *Shape Description from Sparse and Imperfect Data.* PhD thesis, University of Southern California, December 1988. IRIS Technical Report 250.

[18] W. A. Richards and D. D. Hoffman. Codon constraints on closed 2d shapes. *Computer Vision, Graphics and Image Processing*, 31:265–281, 1985.

[19] H. Rom and G. Medioni. Hierarchical decomposition and axial representation of shape. In *Proceedings of the SPIE conference on Geometric Methods in Computer Vision*, San Diego, California, July 1991.

[20] A. Rosenfeld. Axial representations of shape. *Computer Vision, Graphics and Image Processing*, 33:156–173, 1986.

[21] P. Saint-Marc and G. Medioni. B-spline contour representation and symmetry detection. In *Proceedings of European Conference on Computer Vision*, pages 604–606, Antibes, France, 1990.

[22] P. Saint-Marc and G. Medioni. B-spline contour representation and symmetry detection. Technical Report IRIS-262, University of Southern California, Los Angeles, California, February 1990.

[23] F. Ulupınar and R. Nevatia. Inferring shape from contour for curved surfaces. In *Proceedings of International Conference on Pattern Recognition*, pages 147–154, Atlantic City, NJ, 1990.

Reconstructing Surfaces from Unstructured 3D Points*

P. Fua and P. Sander

SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
USA

INRIA Sophia-Antipolis
2004 Route des Lucioles
06565 Valbonne Cedex
France

Abstract

Most active and passive range finding techniques yield unstructured and generally noisy 3D points. In order to build useful world representations, one must be able to remove spurious data points and group the remaining into meaningful surfaces.

In this paper, we propose an approach based on fitting local surfaces. Differential properties of these surfaces are first used iteratively to smooth the points, and then to group them into more global surfaces while eliminating errors.

We present results on complex indoor and outdoor scenes using stereo data as our source of 3D information.

1 Introduction

To reconstruct object surfaces, one can start with a number of measuring techniques, for example laser rangefinding, stereo or 3D scanners, which all provide raw information about the location of points in space. These points, however, often form potentially noisy "clouds" of data instead of the surfaces one expects.

Deriving the surfaces from such data is a difficult task because:

- the 3D points may form a very irregular sampling of the space,
- they may have been produced by several sensors or derived from several viewpoints so that it becomes impossible to work only in the imaging plane of any one sensor,
- several surfaces can overlap — the 2 1/2 D hypothesis required by simple interpolation schemes is not necessarily valid,
- the sensors and algorithms make mistakes that must be properly dealt with.

In this paper, we address the problem of determining surfaces from a set of points in space, where the points are assumed to be nonregular samples from underlying imaged surfaces. Most existing approaches to this problem, such as deformable superquadrics [Terzopoulos and Metaxas, 1990], modal representations [Horowitz and Pentland, 1991] or 3D deformable surfaces [Cohen *et al.*, 1991], assume that all data points belong to a single object to which a model can be fit. In other words, they assume that the data is already segmented, even though it is well known that segmentation is hard when the data is noisy and originates from multiple objects in a scene.

To overcome this problem, we propose using local 3D surfaces to smooth the data points which are then grouped into globals surfaces. We proceed by fitting a local quadric patch to a small neighborhood of each 3D point and using the estimated surfaces to iteratively smooth the raw data. We then use these local surfaces to define binary relationships between points: points whose local surfaces are consistent are considered as related, i.e., sampled from the same underlying surface. Given this relation, we can impose a graph structure upon our data and define the surfaces we are looking for as sets of points forming connected components of the graph. The surfaces can then be interpolated using simple techniques such as Delaunay triangulation. In effect, we are both segmenting the data set and reconstructing the 3D surfaces.

In the next section, we introduce our fitting procedure (mathematical details are left to App. A) and describe our implementation for real 3D data. We then demonstrate our technique using stereo depth images corresponding to complex indoor and outdoor scenes. Our 3D data is produced by a stereo algorithm that has been developed in previous work [Fua, 1991a; Fua, 1991b], and is briefly described in App. B. Note, however, that closely related methods have also been applied to magnetic resonance imagery [Sander and Zucker, 1990] and laser rangefinder images [Ferrie *et al.*,].

2 From Points in Space to Global Surfaces

Our goal is to determine explicit object surfaces from "clouds" of unstructured 3D points. In this section, we describe the implementation of our method and show how it allows us to go all the way from raw data sets to triangulated maps. The procedure consists of the four

*Support for this research was partially provided by ESPRIT P2502 (VOILA) and ESPRIT BRA 3001 (INSIGHT) and a Defense Advanced Research Projects Agency contract.

steps described in the following subsections.

1. Iterative smoothing of the points by local surface fitting.
2. Resampling and merging of the points on a regular 3D grid.
3. Computation of an adjacency graph and clustering of connected components into surfaces.
4. Triangulation of the clusters.

We take the local surfaces to be quadric because they allow curvature computation while having few enough parameters to allow for a reasonably stable fitting process.

Our method is inherently local and parallel; the procedures have been designed with a SIMD architecture in mind and have been implemented on a Connection Machine.[1]

2.1 Local Surfaces

We fit local surfaces by iteratively fitting quadric patches around every data point and then using them to move the points themselves, as shown in Figure 1.

For our algorithm to be effective with real data, it must be:

- Orientation independent: we do not assume any particular orientation of the surfaces, and the result should not be affected by rotations of the objects of interest.
- Insensitive to outliers: In the neighborhood of any point, the possibility of finding points that are in gross error or belong to more than one surface always exist and must be addressed since least-squares techniques are notoriously sensitive to such problems.

Orientation. To achieve orientation independence around a point $P_0 = (x_0, y_0, z_0)$, we use all the points in a spherical neighborhood and estimate the orientation of the local tangent plane. To initialize, we simply fit a plane; thereafter we use the quadric fit of the previous iteration. Given this orientation, we define a reference frame whose origin is P_0 itself and whose z axis is perpendicular to the plane; we fit a quadric of the form

$$z = \text{quad}(x, y) = ax^2 + bxy + cy^2 + dx + ey + f \quad , \quad (1)$$

by minimizing a least squares criterion δ,

$$\delta = \sum_i w_i (z_i - \text{quad}(x_i, y_i))^2 \quad , \quad (2)$$

where the $(x_i, y_i, z_i)_{1 \le i \le n}$ are the n neighbors of P_0 and the w_i are associated weights. We then transform P_0 as follows:

$$P_0 - - > (0, 0, f = quad(0, 0)) \quad (3)$$

expressed in the local reference frame.

In effect, we are approximating the 2nd order Taylor expansion of the surface around P. Since we iterate the estimates, this procedure is a form of relaxation (see Appendix A) where the amount of smoothing increases with the number of iterations and the size of the neighborhoods. However, because we use quadric patches as opposed to planar ones, the procedure preserves curvature and does not smooth out relevant features. For typical stereo data sets ($\approx$ 100,000 points), the algorithm converges within five to ten iterations.

Outliers. To deal with outliers, we define a metric d_{quad} that measures whether or not two points appear to belong to the same surface. We take d_{quad} to be

$$d_{quad}(P_1, P_2, quad_1, quad_2) = \max(dist_1, dist_2) \quad (4)$$

where

$$\begin{aligned} dist_1 &= abs(z_1 - quad_2(x_1, y_1)) \\ &\quad \text{expressed in the reference frame of } quad_2 \\ dist_2 &= abs(z_2 - quad_1(x_2, y_2)) \\ &\quad \text{expressed in the reference frame of } quad_1 \ . \end{aligned}$$

We depict d_{quad} graphically in Figure 2. It is zero when the two points belong to the same local surface and increases when their respective local surfaces become inconsistent. It can therefore be used to discount outliers by computing the weighting factor w_i of equation 2 at iteration t to be:

$$\begin{aligned} w_i &= \frac{1}{1 + (dist_i/\sigma)^2} \quad (5) \\ dist_i &= d_{quad}(P_0, P_i, \text{quad}_0^{t-1}, \text{quad}_i^{t-1}) \end{aligned}$$

where the $(quad_i^{t-1})_{0 \le i \le n}$ are the quadrics that had been computed at the previous iterations and σ is an estimate of the variance of the process generating the data points. Note that for processes such as stereo compilation or laser range finding, σ can actually be estimated.

In this manner, as the algorithm progresses, the points that are on the same surface as P_0 gain influence while the others are increasingly discounted. The discriminatory power of the algorithm is substantially increased by fitting the quadrics several times at every iteration and updating the weights without moving the data points. We illustrate this behaviour in Figure 3. The two noisy hemispheres are smoothed without being merged and the points between them are left as outliers; such a result would be difficult to achieve with a simple 2 1/2D interpolation scheme.

Implementation. The main obstacle to performing the relaxation smoothing is the very large amount of data that has to be taken into account. A dense 512x512 depth map represents over 250,000 points and as many local surfaces, and the problem obviously gets worse as more and more views are being merged. One solution is to merge the points as they are acquired [Szeliski, 1990]. In this work, however, we are interested more in the behaviour of the smoothing algorithm than in the possible ways of reducing the amount of computation. We therefore start with *all* the data points and assume that, in the z direction, at most a limited number of surfaces can overlap. This is typically true for ground level scenes if z is the vertical direction because only a finite (and usually small) number of objects are stacked on top of one another. Given this assumption, we can

[1] Trademark of Thinking Machines Inc.

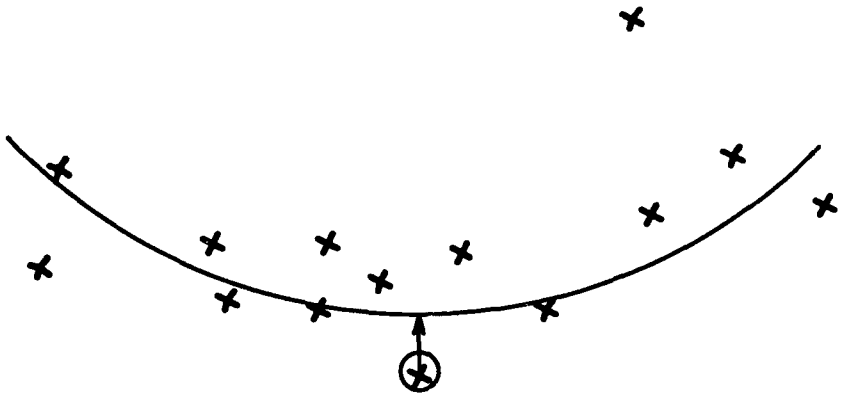


Figure 1: Local surface fitting: the neighbors of the circled point are used to fit a quadric surface onto which the point is then projected.

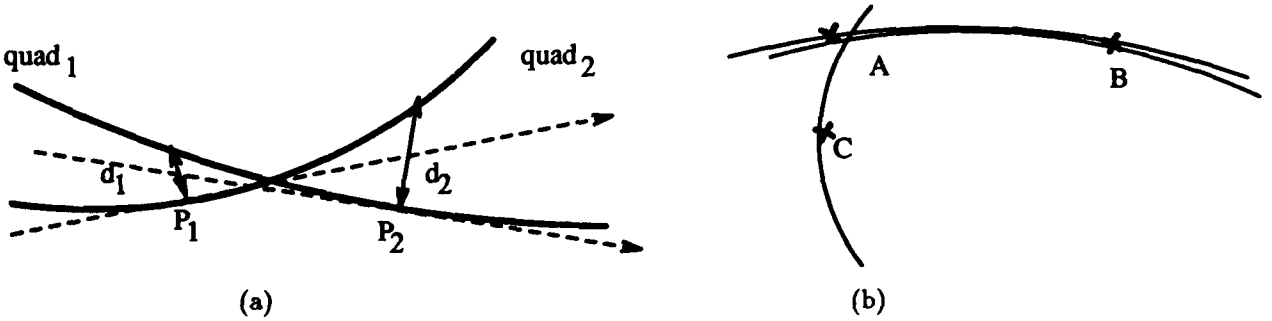


Figure 2: (a) The distance between two points is taken to be the maximum value of the distance of one point to the local surface corresponding to the other, represented by the arrows. The dotted lines represent the axes of the reference frames in which the computations are performed. (b) For this metric, point B is "close" to A but C is not, even though their euclidean distances are comparable.

create a cube-shaped data structure by quantizing the x and y coordinate axes and stacking the 3D points with the same quantized x, y values in columns ordered by increasing z values. In this way, we can guarantee that all neighbors can be found in a cubic neighborhood around every point in the data structure; this allows efficient 4-connected (NEWS) access to the neighbors. We can also reduce the z dimension of the cube data structure and make efficient use of the Connection Machine processors.

2.2 Resampling

When the smoothing is done, as shown in Figure 3(c), the data points still form an irregular sampling of the underlying surfaces that is ill suited for the generation of a map. However, to every point is associated a local surface defined by the quad function of equation 1. In order to produce meaningful triangulations, we need a more regularly spaced set of vertices. We therefore pick spatial step sizes δ_x,δ_y and δ_z along an the X,Y and Z axes of an absolute referential. For points whose local surface patch has a normal within 45 degrees of the Z axis, we use the following updating scheme

$$\begin{aligned} x &\dashrightarrow x_0 = \delta_x \mathrm{mod}(x, \delta_x) \\ y &\dashrightarrow y_0 = \delta_y \mathrm{mod}(y, \delta_y) \\ z &\dashrightarrow \mathrm{quad}(x_0, y_0). \end{aligned}$$

For points whose normals are closer to the X or Y axes, we use the same method but permute the roles of x,y and z. Furthermore, after updating, several points may have two identical coordinates and, provided that their third ones are close enough, be merged. If these points have been seen in different views, we retain that information for later use as will be discussed in Section 3.1. In this manner, points are "resampled" on a regular 3D grid as can be seen in Figure 3(d).

In effect we are replacing a large number of irregularly spaced 3D points by a smaller set of regularly spaced ones and their local surfaces: we are achieving both data organization and compression. This turns out to be an effective way to merge data-points coming from several views or sensors.

2.3 Clustering

To cluster the isolated 3D-points into more global entities, we use again the metric of Equation 4 to define a "same surface" relationship $\mathcal{R}$ as follows:

$$pt_1 \mathcal{R} pt_2 \iff \left| \begin{array}{lcl} d_{quad}(pt_1, pt_2) & < & \max_q \\ d_{\mathrm{eucl}}(pt_1, pt_2) & < & \max_d \end{array} \right. \quad (6)$$

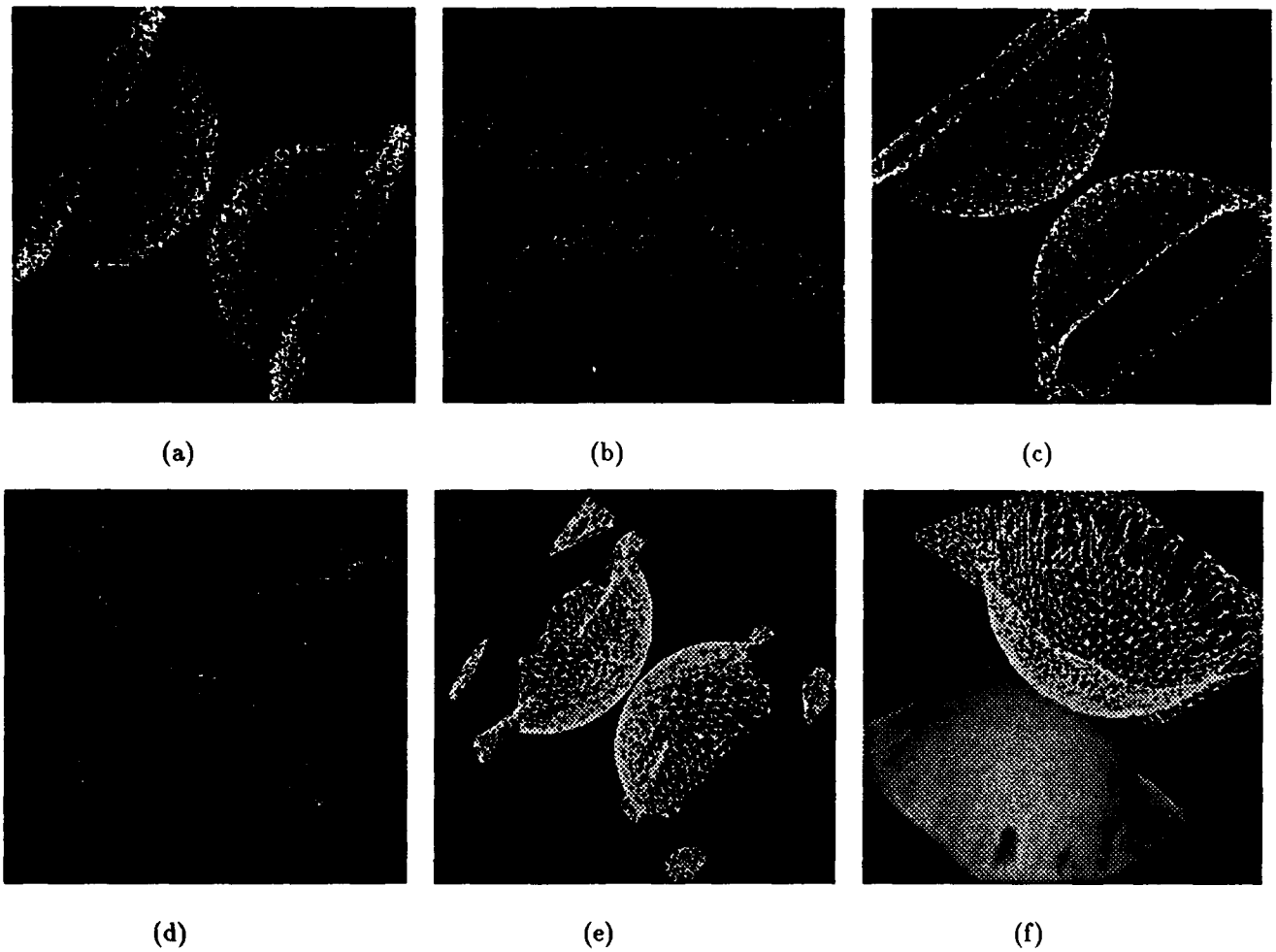


Figure 3: (a) Two superposed and noisy hemispheres. (b) The top of the closest parts of the spheres. (c) Smoothed spheres after several iterations. (d) Resampled points. (e) (f) Two possible segmentations of the data points for two different values of the max_q parameter of Equation 6. In both cases the clusters of points have been triangulated using a 2D Delaunay triangulation in the horizontal plane and in (f) one of them has been shaded.

where d_{eucl} is the euclidean distance max_d and max_q are two thresholds. In other words, two points are assumed to belong to the same surface if their local fits are consistent with one another.

The data set equipped with the relationship $\mathcal{R}$ can now be viewed as a graph whose connected components are the surfaces we are looking for. In practice, there may be erroneous points in the original range data, resulting in situations like the one shown in Figure 4, where legitimate clusters are weakly linked. In such cases, we have found that removing all points that do not have a minimum number of neighbors allows us to throw away the gross errors and generate meaningful clusters. In the case of the two hemispheres of Figure 3, depending on the value the max_q threshold of Equation 6, the algorithm finds either the two obvious clusters or separated the hemispheres themselves and their flat bases because of the sudden change of orientation at the junctions.

2.4 Triangulating

The clusters we have generated so far are collections of points and their associated local surfaces. For many applications, such as robotics or graphics, it is important to be able to unambiguously interpolate the surfaces. Delaunay triangulation is an excellent way of doing this and, furthermore, lets us compute shaded models of our data sets. Alternatively, because our data points are now grouped into sets of points that belong to the same global surface, we could use a number of the other published methods [Cohen *et al.*, 1991; Horowitz and Pentland, 1991; Terzopoulos and Metaxas, 1990].

For applications in which the global surfaces can be projected onto a plane without losing their topology, we do so and use 2D Delaunay triangulation in that plane. This is typically the case for mobile robotics when one needs to model the world as a ground level surface plus obstacles. If this ground level surface has no overhangs, it can be projected onto the horizontal plane. The triangulations of Figure 3(e) and (f) have been computed

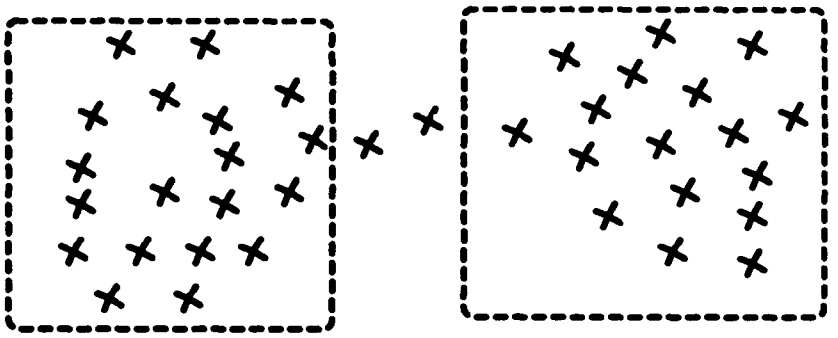


Figure 4: Two legitimate surfaces are can be weakly linked by a few outliers.

in this manner.

For applications where the surfaces can have any orientation, we perform a full 3D Delaunay triangulation that yields the convex hull of the points and use the faces of the tetrahedrons as our surface primitives.

In both cases we retain only the triangles whose orientation is consistent with the local surface patches of the vertices. In the current implementation, we test only the consistency of the surface normals with the triangle orientation, as shown in Figure 2.4.

When the 3D data is sufficiently clean, as will be shown in the results section, this heuristic is sufficient to remove the spurious triangles and "hollow out" the triangulation. However, in more difficult cases, more powerful constraints should be brought to bear:

- Visibility constraint: If a point is visible in a given view but appears to be behind one of the triangles, one of the two must be spurious. Since the resampling operation described in section 2.2 provides such information, it ought to be used.
- Supporting evidence: Given an estimate of the variance of the process generating the points, the σ of Equation 5, one can estimate how much data supports the existence of each triangle.

These extensions will be be the subject of future work.

3 Reconstructing 3D Surfaces from Stereo Data

In this section, we show how our technique can be used to reconstruct 3D surfaces using stereo depth-maps. We first reconstruct natural scenes by merging several maps and then demonstrate the potential of our algorithm on a complex 3D scene containing stacked-up man-made objects.

3.1 Mergeing Results Computed from Several Viewpoints

The stereo algorithm we use [Fua, 1991a; Fua, 1991b] produces semi-dense maps that are 2 1/2D representations of the world. They are necessarily incomplete and, in particular, cannot account for the occluded parts of the scene, such as the back of the rocks in Figure 6(a).

To reconstruct the scene more completely, we have used 5 sets of stereo-pairs corresponding to camera positions between that of figures 6(a) and 6(c). After having registered the stereo results with one another [Zhang and Faugeras, 1990], we can use our technique to generate clusters of triangulated 3D points. The largest one, depicted in Figure 7, accounts for all the large rocks. The erroneous data points have been discarded as outliers.

In Figure 8, we use two stereo pairs to reconstruct the ground surface. Note that the raw points corresponding to the tree and the background wall, as well as the erroneous matches, have been segmented out.

In the two examples above, the depth maps were quite dense and the final surface of interest could be projected onto the horizontal plane to compute a 2D Delaunay triangulation. In the next subsection, we show a more difficult case for which none of these conditions holds.

3.2 Reconstructing a Complex 3D Scene

In Figure 9, we show three objects that have been stacked on a turntable. By rotating the table, we have produced 12 stereo triplets and their corresponding depth maps such as the one of Figure 9 (c). Because the objects are not very textured, the individual maps are neither very dense nor very precise; however by merging them using our smoothing/resampling procedure we can generate the set of points shown in Figure 9(d) and (e) that correctly captures the geometry of the main structures. Unfortunately, there also are a few erroneous points that seem to float in space that must be removed. The shaded models of Figure 10 have been obtained by computing a 3D Delaunay triangulation of the points and retaining only the faces whose orientation is consistent with the local surfaces, as depicted in Figure 2.4. The surfaces corresponding to the three objects are clearly separated but still exhibit a few "holes" due to the lack ot stereo data.

4 Conclusion

In this paper we have presented a method for estimating explicit surfaces from given data consisting of a nonregular sampling of the surfaces. The data was assumed in the form of a general "cloud of points" with no particular structure, and we made no *a priori* assumptions about the form of the underlying surfaces, other than

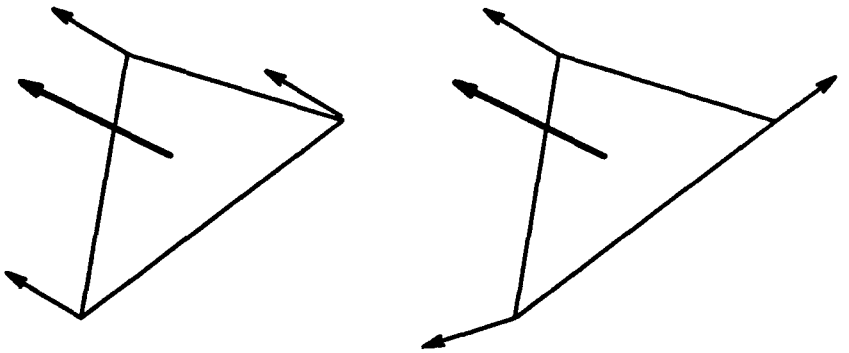


Figure 5: (a) A valid triangle: the normal at the vertices are consistent with the triangle orientation. (b) An invalid triangle.

smoothness. We have shown that we can effectively reconstruct global surfaces under such difficult conditions. However, when additional assumptions or knowledge can be brought to bear, one should obviously do so and our technique can then be used as a source of information about the local surface geometry.

The algorithm consists of four sequential steps: smoothing of the points by iterative local surface fitting; resampling the smoothed points onto a regular grid; computation of an adjacency graph of the points with clustering of the connected components; and triangulation of the clusters. The algorithm is well-suited to parallel implementation and the examples shown were produced on the Connection Machine.

The experiments shown in this paper used depth data acquired from stereopsis by a mobile robot, although we have successfully tested much of the same code on 3D biomedical scanner images as well. It must ne noted that when the quality of the data degrades, the critical step becomes the grouping one. In extreme cases, the heuristics described in this paper are too simple and may fail. To push the nethod forward, it will be necessary to develop more sophisticated ones. In fact, generating the triangulations of section 2.4 could be recast as the problem of finding the best description of the data in terms of a set of triangles knowing the normals and curvatures and every vertex. This problem can be handled within the framework provided by Mininum Description Length encoding [Rissanen, 1987; Leclerc, 1989; Fua and Hanson, 1989]. Such a framework sould provide us with a sound theoretical basis for future work.

Appendices

A The mathematics of recursive fit-and-update

In this appendix, we give some of the details of the method of fitting local surfaces. A comprehensive analysis of the mathematical properties of the algorithm is not given, but will appear in a future publication. Also, to simplify the presentation, we reduce the problem by one dimension and consider fitting local quadratic curves to 2D data points.

Given m initial data points $\{(x_i^0, y_i^0),\ i = 1, \ldots, m\}$, we assume that the iterations k take place on a fixed regular grid so that $x_i^{k+1} = x_i^k = \cdots = x_i^0 = x_i$. We further assume that the fits are local and over the same size neighborhoods of n points so that the neighborhood of (x_i, y_i^k) consists of $\{(-\frac{n-1}{2}, y^k_{i-\frac{n-1}{2}}), \ldots, (0, y_i^k), \ldots, (\frac{n-1}{2}, y^k_{i+\frac{n-1}{2}})\}$. The initial fits to the data points determine coefficients $\zeta_i^0 = (a_i^0, b_i^0, c_i^0)$ of the "best-fitting" quadratic

$$y_i^0(t) = a_i^0 t^2 + b_i^0 t + c_i^0$$

at (x_i, y_i^0).

We now look at the relationship between successive iterations k and $k-1$. The quadratic fit at (x_i, y_i^k) is the least-squares solution of the overdetermined system $A_i^k \zeta_i^k = \beta_i^k$, and for simplicity we consider here that it is given by

$$\zeta_i^k = (A^{\mathrm{T}} A)^{-1} A^{\mathrm{T}} \beta_i^k.$$

Note that A_i^k is composed only of linear combinations of x_j and is hence independent of k, and since the $\{x_j\}$ in each neighborhood are the same, it is independent of i as well and we write just A. Thus,

$$\begin{pmatrix} a_i^k \\ b_i^k \\ c_i^k \end{pmatrix} = (A^{\mathrm{T}} A)^{-1} A^{\mathrm{T}} \begin{pmatrix} y^k_{-\frac{n-1}{2}} \\ \vdots \\ y^k_{\frac{n-1}{2}} \end{pmatrix},$$

where $(a_i^k, b_i^k, c_i^k) = \zeta_i^k$. Now, the data point y_i^k is updated to $y_i^{k+1} = c_i^k$, and introducing this updating into the above equation gives, at the next iteration,

$$\begin{aligned} \zeta_i^{k+1} &= \begin{pmatrix} a_i^{k+1} \\ b_i^{k+1} \\ c_i^{k+1} \end{pmatrix} = (A^{\mathrm{T}} A)^{-1} A^{\mathrm{T}} \begin{pmatrix} y^{k+1}_{-\frac{n-1}{2}} \\ \vdots \\ y^{k+1}_{\frac{n-1}{2}} \end{pmatrix} \\ &= (A^{\mathrm{T}} A)^{-1} A^{\mathrm{T}} \begin{pmatrix} c^k_{-\frac{n-1}{2}} \\ \vdots \\ c^k_{\frac{n-1}{2}} \end{pmatrix}. \end{aligned}$$

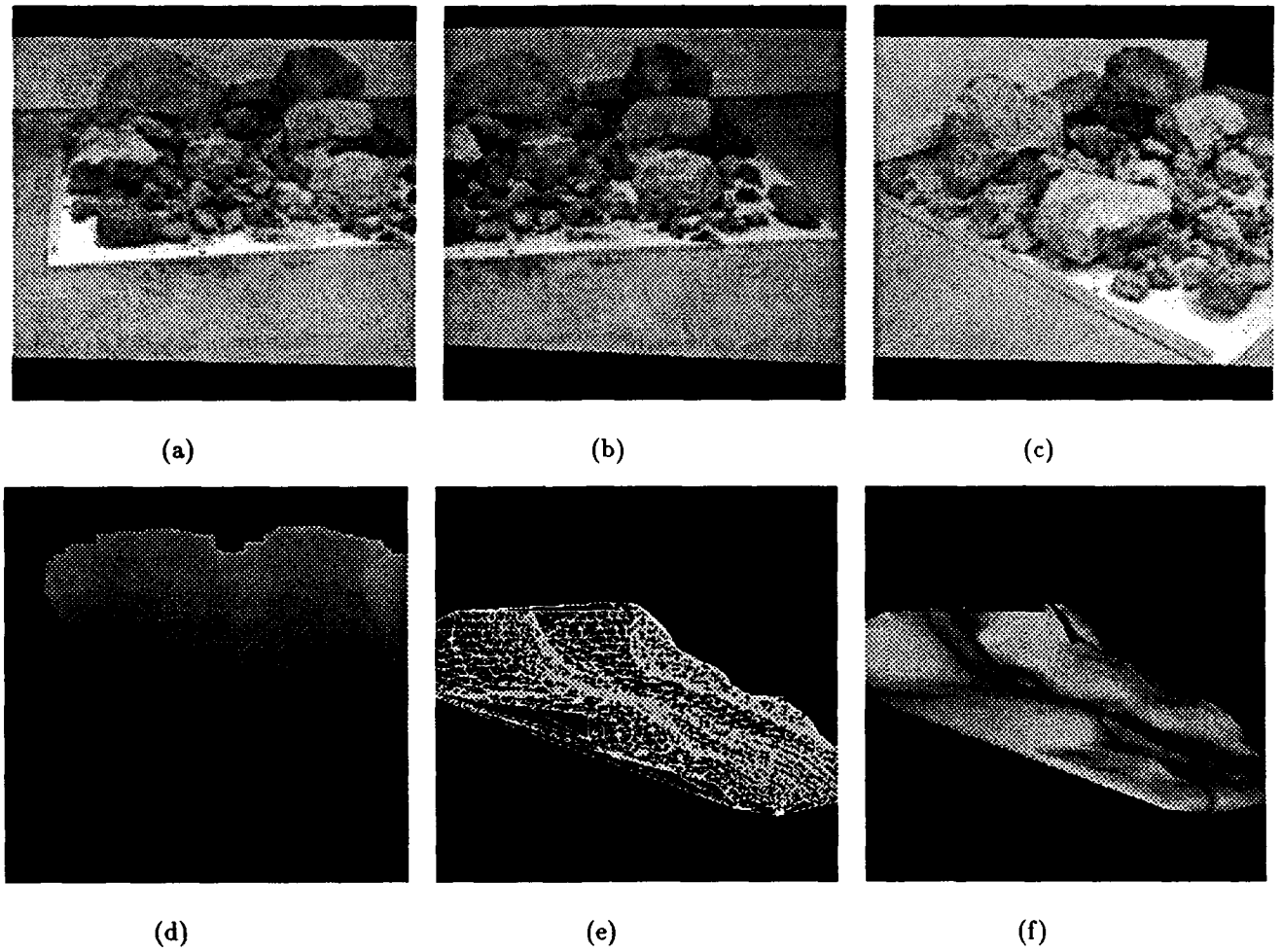


Figure 6: (a) (b) A stereo pair showing a set of rocks. (c) Another image taken from a completely different viewpoint. (d) The stereo map derived by matching (a) and (b). The black areas correspond to textureless areas for which no depth was computed. Elsewhere, the lighter colors correspond to the greater distances (e) Wireframe representation. (f) Shaded representation. Note that the parts of the rocks visible in both (a) and (b) are correctly reconstructed, but that the others are not.

Now, when only the third component of ζ_i^k is considered, we obtain

$$c_i^{k+1} = \left\{(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}\right\}_{[3,\bullet]} \begin{pmatrix} c^k_{i-\frac{n-1}{2}} \\ \vdots \\ c^k_{i+\frac{n-1}{2}} \end{pmatrix} = L \begin{pmatrix} c^k_{i-\frac{n-1}{2}} \\ \vdots \\ c^k_{i+\frac{n-1}{2}} \end{pmatrix},$$

where $B_{[3,\bullet]}$ is the third row of a matrix B and L is a $1 \times n$ matrix. Now, we see that considering *all* the data points at iteration $k+1$, we get

$$\begin{pmatrix} \vdots \\ c^{k+1}_{i-1} \\ c^{k+1}_{i} \\ c^{k+1}_{i+1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \ddots & & & & \\ & L & & & \\ & & L & & \\ & & & L & \\ & & & & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ c^k_{i-1-\frac{n-1}{2}} \\ \vdots \\ c^k_{i-1+\frac{n-1}{2}} \\ c^k_{i-\frac{n-1}{2}} \\ \vdots \\ c^k_{i+\frac{n-1}{2}} \\ c^k_{i+1-\frac{n-1}{2}} \\ \vdots \\ c^k_{i+1+\frac{n-1}{2}} \\ \vdots \end{pmatrix},$$

where each L matrix occupies a $1 \times n$ block of the large matrix. By shifting the L matrices appropriately, the right-hand side can be written more economically as the



(a) (b) (c)

Figure 7: (a) Triangulated ground surface for the rocks of Figure 6 (b) (c) Shaded views. Note that the backs of the foreground rocks of Figure 6(a) are now clearly visible.

product of an $m \times m$ matrix and an $m-$vector,

$$\begin{pmatrix} \vdots \\ c_{i-1}^{k+1} \\ c_i^{k+1} \\ c_{i+1}^{k+1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \ddots & & & & \\ & L & & & \\ & & L & & \\ & & & L & \\ & & & & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ c_{i-1}^{k} \\ c_i^{k} \\ c_{i+1}^{k} \\ \vdots \end{pmatrix}$$

$$= \begin{pmatrix} \ddots & & & & \\ & L & & & \\ & & L & & \\ & & & L & \\ & & & & \ddots \end{pmatrix}^{k+1} \begin{pmatrix} \vdots \\ c_{i-1}^{0} \\ c_i^{0} \\ c_{i+1}^{0} \\ \vdots \end{pmatrix}$$

$$= M^{k+1} \begin{pmatrix} \vdots \\ c_{i-1}^{0} \\ c_i^{0} \\ c_{i+1}^{0} \\ \vdots \end{pmatrix},$$

where M is an $m \times m$ banded matrix of width n. Thus we see that with a bit of work the results of the quadratic fit at iteration k can be derived from the results of the initial fit, in turn obtained from the given data observations.

B Area Based Stereography

In this section, we briefly describe the correlation-based stereo algorithm used to produce the 3D data we need. For a more complete description we refer the interested reader to previous publications [Fua, 1991a; Fua, 1991b].

Most correlation based algorithms attempt to find points of interest on which to perform the correlation. This approach is justified when only limited computing resources are available, but with modern hardware architectures and massively parallel computers it becomes possible to perform the correlation over all image points and retain only matches that appear to be "valid."

To generate a dense and accurate depth map, one must then interpolate these measures in such a way as to preserve depth discontinuities. To do so, we model the world as made of smooth surfaces separated by depth discontinuities that generate changes in grey level intensities due to changes in orientation and surface material.

Given a pair of images and corresponding camera models, the computation of the depth map consists of four steps.

1. **Rectification.** The images are reprojected onto the same image plane so that all epipolar lines become parallel. This makes the parallel implementation of the correlation algorithm much simpler because the exact same operations are performed at every pixel.
2. **Matching.** Correlation scores are computed by comparing a fixed window in the first image to a shifting window in the second. The second window is moved in the second image by integer increments and an array of correlation scores is generated for integer disparity values. To compute the disparity with subpixel accuracy, we fit a second degree curve to the correlation scores in the neighborhood of the maximum and compute the optimal disparity by interpolation.

   As shown by Nishihara [Nishihara and T.Poggio, 1983], the probability of a mismatch goes down as the size of the window and the amount of texture increase. However, using large windows leads to a loss of accuracy and the possible loss of important image features. We choose to consider as acceptable only those results for which we get the same result by reversing the roles of the two images, thereby greatly reducing the probability of error even when using very small windows (down to 3x3 for textured outdoor scenes). In fact the density of such consistent matches in a given area of the image appears to be an excellent indicator of the quality of the stereo matching. An occasional "false positive" (a pixel for which the same erroneous disparity is measured when matching both from left to right and right to left) may occur, but we have never encountered a

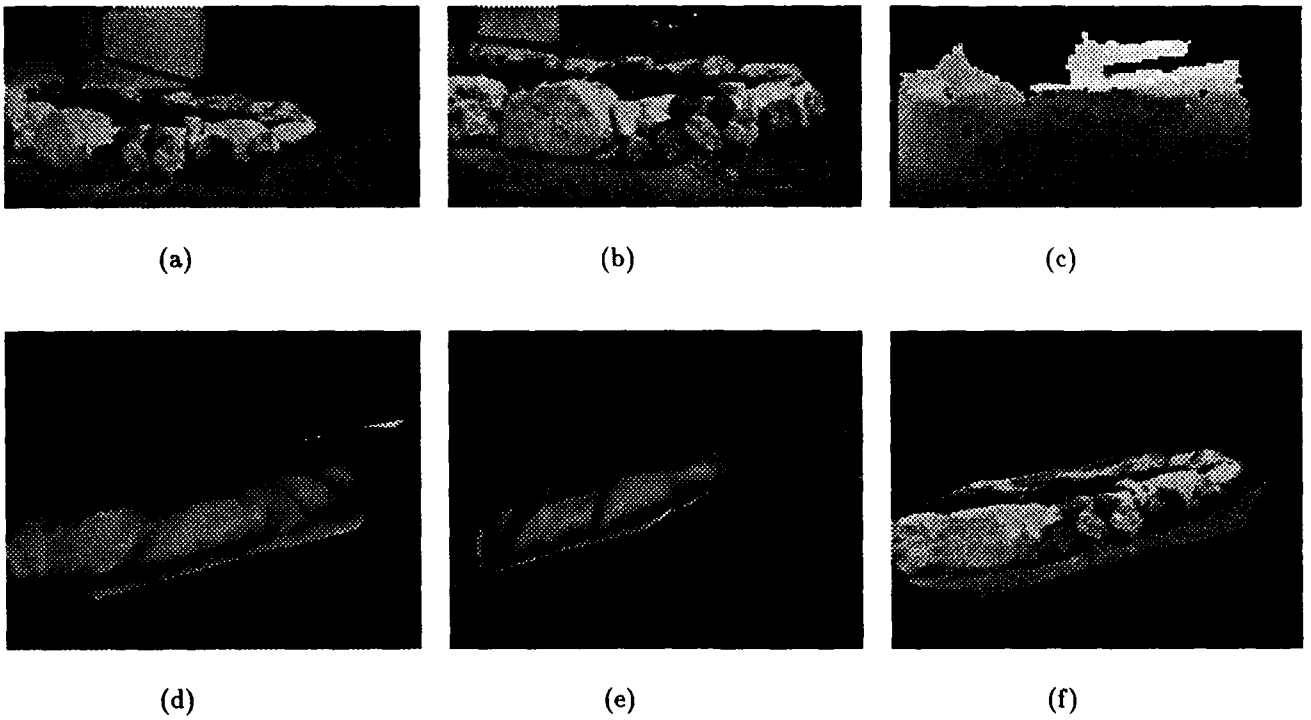


Figure 8: (a) (b) The two left images of a pair of stereo views. (c) The depth map computed using the first one. (d) (e) Two shaded views. (f) A texture mapped view.

situation that gave rise to a large clump of such errors.

3. **Merging across Hierarchy.** We perform the matching at several levels of resolution with identical window sizes, which is conceptually equivalent to matching at one level with windows of different sizes [Kanade and Okutomi, 1990] but computationally more efficient. We then pick the disparity computed at the highest level of resolution for which an acceptable disparity with respect to our consistency test can be found.

   This is a departure from traditional hierarchical implementations that make use of the results generated at low resolution to guide the search at higher resolutions. While these are good methods for reducing computation time, they assume that the result generated at low resolution are more reliable, even if less precise, than those generated at high resolution. This is a questionable assumption especially in the presence of occlusions.

4. **Interpolation.** Finally, the dense depth map w is computed by fitting a piecewise smooth surface to the correlation-based depths $w0$. Assuming that depth discontinuities are more likely to occur where the image intensity gradient is large, we use a conjugate gradient method [Szeliski, 1990; Terzopoulos, 1986] to minimize the following criterion:

$$C = c(w - w0)^2 + \lambda_x \frac{\partial w}{\partial x}^2 + \lambda_y \frac{\partial w}{\partial y}^2 ,$$

   where c is zero if the correlation has failed and proportional to the normalized correlation score otherwise, and λ_x, λ_y are inversely proportional to the image gradients in the x and y directions. This interpolation scheme produces dense depth maps that preserve depth discontinuities without having to explicitly specify their location.

This algorithm uses simultaneously the grey level information present in a single image and the stereo information present in a pair. In fact, as suggested by Moravec [Moravec, 1981] and many others, more than two images can and should be used whenever practical.

However, the smoothing scheme described above operates in the image plane and makes the 2 1/2D assumption of all such interpolation techniques. In particular, it cannot deal with occlusions and hidden faces. To recover the full 3D structure of objects and merge range data computed from several viewpoints we need to invoke the 3D smoothing procedure of Section 2.

References

[Cohen *et al.*, 1991] Isaac Cohen, Laurent Cohen, and Nicholas Ayache. Introducing deformable surfaces to segment 3D images and infer differential structure. Technical report, INRIA, 1991.

[Ferrie *et al.*,] Frank P. Ferrie, Jean Lagarde, and Peter Whaite. Recovery of volumetric object descriptions from laser rangefinder images. These proceedings.

[Fua and Hanson, 1989] P. Fua and A.J. Hanson. Objective functions for feature discrimination. In *IJCAI-89 Conference*, Detroit, August 1989.

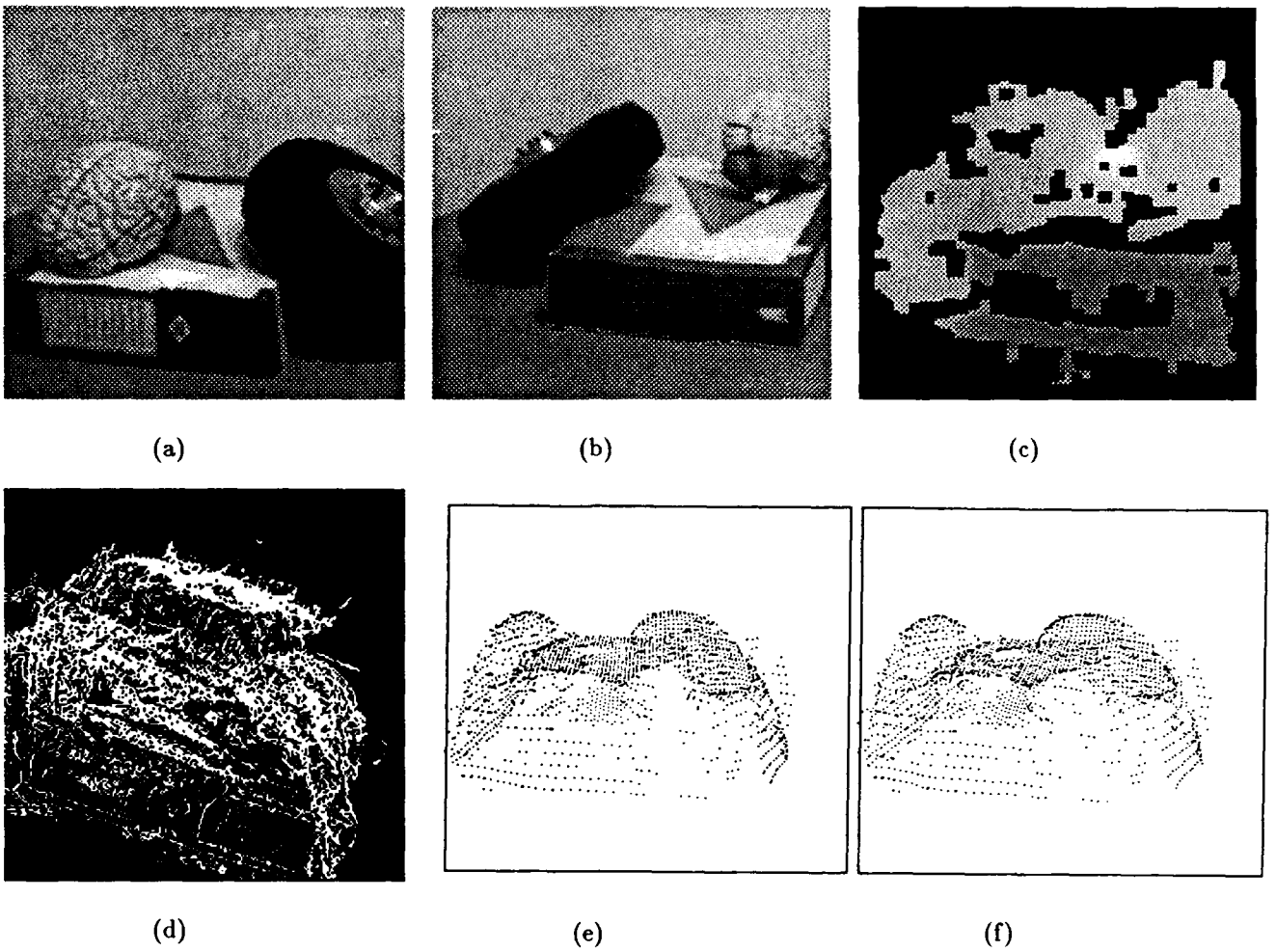


Figure 9: (a) (b) Two diametrically opposing views of three objects on a turntable: a box, the wheel of a robot, and the brain of an unsuccessful graduate student. (c) The depth map corresponding to viewpoint b. (d) The raw stereo data from all views shown as a noisy cloud of points. (e) (f) The smoothed and resampled points shown as a crossed-eye stereogram to be fused by the reader. Note that the vertical sides and the horizontal top of the box, the wheel, and the brain are correctly recovered.

[Fua, 1991a] P. Fua. Combining stereo and monocular information to compute dense depth maps that preserve depth discontinuities. In *IJCAI Conference*, Sydney, Australia, August 1991.

[Fua, 1991b] P. Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 1991. Accepted for publication.

[Horowitz and Pentland, 1991] Bradley Horowitz and Alex Pentland. Recovery of non-rigid motion and structure. In *Proceedings of CVPR'91*, pages 325–330, Hawaii, June 1991.

[Kanade and Okutomi, 1990] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptative window: Theory and experiment. In *Image Understanding Workshop*, September 1990.

[Leclerc, 1989] Y. G. Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision*, 3(1):73–102, 1989.

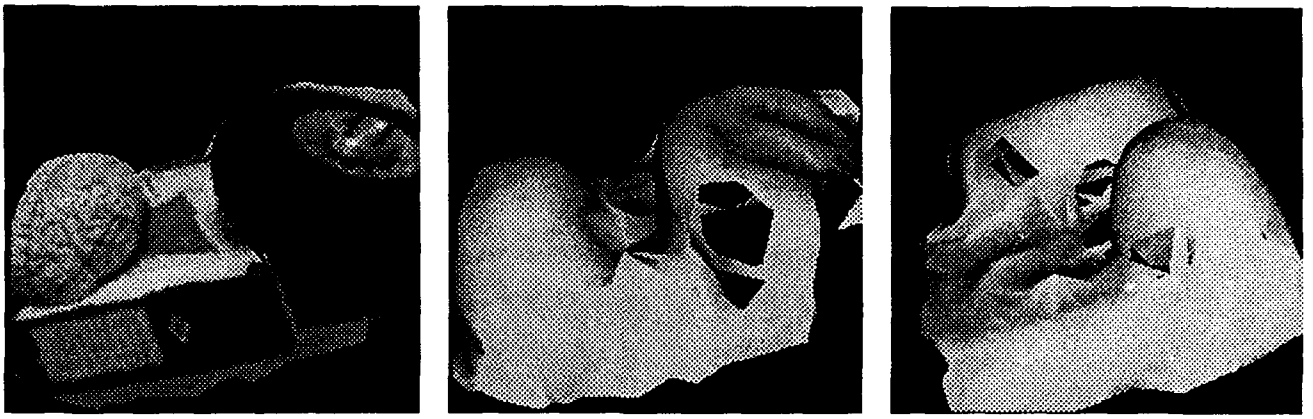
[Moravec, 1981] H. Moravec. *Robot Rover Visual Navigation*. UMI Research Press, Ann Arbor, Michigan, 1981.

[Nishihara and T.Poggio, 1983] H.K. Nishihara and T.Poggio. Stereo vision for robotics. In *ISRR83 Conference*, Bretton Woods, New Hampshire, 1983.

[Rissanen, 1987] J. Rissanen. Minimum-description-length principle. *Encyclopedia of Statistical Sciences*, 5:523–527, 1987.

[Sander and Zucker, 1990] Peter T. Sander and Steven W. Zucker. Inferring surface trace and differential structure from 3-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(9):833–854, September 1990.

[Szeliski, 1990] R. Szeliski. Fast surface interpolation using hierarchical basis functions. *IEEE Transac-*



(a) (b) (c)

Figure 10: The triangular faces of 3D Delaunay triangles whose orientations are consistent with the local surface patches. (a) Texture mapped view. (b) (c) Shaded views. The texture maped view is obviously more realistic but also much less revealing of the actual performance of the algorithm. Note also, that the algorithm leaves holes where there was not enough stereo data to build reliable triangles.

tions on Pattern Analysis and Machine Intelligence, 12(6):513–528, June 1990.

[Terzopoulos and Metaxas, 1990] Demetri Terzopoulos and Dimitri Metaxas. Dynamic 3d models with local and global deformations: Deformable superquadrics. In *Proceedings of ICCV'90*, pages 606–615, Osaka, December 1990.

[Terzopoulos, 1986] D. Terzopoulos. Image analysis using multigrid relaxation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):129–139, March 1986.

[Zhang and Faugeras, 1990]
Z. Zhang and O.D. Faugeras. Tracking and motion estimation in a sequence of stereo frames. In L.C. Aiello, editor, *Proc. 9th European Conf. Artif. Intell.*, pages 747–752, Stockholm, Sweden, August 1990.

Recovery of 3-D Objects with Multiple Curved Surfaces from 2-D Contours *

Fatih Ulupınar and Ramakant Nevatia
Institute for Robotics and Intelligent Systems
School of Engineering
University of Southern California
Los Angeles, California 90089-0273
ulupinar@iris.usc.edu, nevatia@iris.usc.edu

Abstract

Inference of 3-D shape from 2-D contours in a single image is an important problem in machine vision. Often, techniques to solve this problem examine each surface in the scene separately whereas our perception of their shapes clearly depends on the interplay between them as well. In this paper, we describe a technique that attempts to recover the shapes of all the surfaces of an object simultaneously, though, it is limited to objects made of zero-Gaussian curvature surfaces. Our technique is based on an analysis of three kinds of symmetries defined in the paper and the constraints that derive from them, and from other boundaries. Results on some complex examples are shown.

1 Introduction

One of the basic goals of mid-level vision is to recover the local orientations of the surfaces of the objects in a scene. Of the many cues available to aid in this process, we believe that shape of the 2-D contour itself is the most important and robust one. Of course, inferring shape from contour is highly ambiguous and can not be done without making some assumptions. The goal in shape from contour methods is to minimize the number of needed assumptions and to achieve results consistent with human perception. For shape from contour analysis, the only *ground truth* is really in human perception, for even if the given contour was obtained by a real object, it *could* have been also obtained by any number of other objects as well.

Early work on inferring 3-D structure from a 2-D shape was focused on analysis of line drawings of polyhedra [Huf71], [Clo71], [Mac73], [Kan81], [KK83]. In the 80s, several techniques for non-polyhedral shapes were proposed (e.g. [BT81], [Wei88], [BY84], [Ste81], [XT87], [HB88], [PCM89], [Nal87]). One characteristic of most of these methods is that they examine only a single surface in the scene at a time whereas our perception of a surface can be strongly influenced by our perception of the entire object.

In earlier papers, we have examined the recovery of 3-D surface shape of a variety of curved surfaces cut by planes ([UN90a], [UN90b], [UN91b]). Our techniques rely on observed symmetries in the image and the analysis depends on the interplay between constraints imposed by the curved surface and the planes cutting it. We showed successful shape recovery for the following kinds of surfaces: zero-Gaussian curvature surfaces [UN90a], surfaces of straight homogeneous generalized cylinders [UN90b] and surfaces of planar, right, constant cross-section generalized cylinders [UN91b].

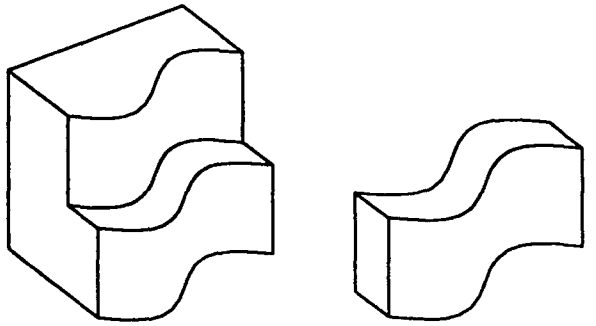


Figure 1: (a) an object consisting of multiple planar and curved surfaces and (b) the front part of the object in isolation.

Complex objects, however, are composed of a number of curved surfaces and planar patches. Our perception of each of these surfaces is affected by the presence of the others. For example consider the object in figure 1 (a) which appears to be a composite of two objects, one in front of the other. If the front object is viewed in isolation as in figure 1 (b), the interpretation for its top surface is ambiguous (it may be planar or not). However, in the context of the whole object (a), this ambiguity disappears (the said surface must be planar). This is an example of how a remote surface changes the perception of some other surface drastically. In general, even if the perception of a surface may not be effected by the neighboring surfaces this drastically, its perception would still be effected by small amounts to make the whole object more consistent (surfaces obeying inter-surface constraints).

This paper explores 3-D surface inference by including interplay between many surfaces that may comprise a complex object. Our technique is limited to a combination of zero-Gaussian curvature (ZGC) surfaces and planar surfaces. However, our method uses constraints similar to those used in our earlier work on shape from

*This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-90-C-0078. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

contour for ZGCs [UN90a]. The previous work, was limited to analysis of a ZGC surface cut by two parallel planes. In this paper, we first develop techniques that allow analysis of a ZGC surface cut by non-parallel planes and then show how multiple ZGC surface shapes can be recovered simultaneously while influencing each other. To accomplish this, we have found it more convenient to change some of the representations used in our earlier work as well as to device an additional form of symmetry.

In section 2 we define three kinds of symmetries and discuss the occurrence of these symmetries in the context of planar and ZGC surfaces. In section 3 we discuss the constraint equations that are used in the shape recovery. In section 4 the representation of the surfaces and single surface recovery is discussed. In section 5 combined shape recovery of multiple surfaces is discussed in detail. In section 6 we discuss an implementation of the shape recovery algorithms and present some results.

Our method assumes that clean, closed boundaries are given (or can be extracted from the real image). We do not address the issue of separating object boundaries from surface markings, or other *perceptual grouping* operations here, though we believe that constraints required for shape inference by our methods will aid in the perceptual organization process itself. Such research is being currently pursued in our laboratory separately.

We assume orthographic projection throughout the paper unless specifically mentioned otherwise (in a separate paper, we have shown how many of the constraints for orthographic projection can be transformed to the case of perspective projection [UN91a].

2 Surfaces and Symmetries

In this paper we concentrate on shape from contour for objects composed of planar and zero Gaussian Curvature surfaces. A Zero Gaussian Curvature (ZGC) surface is one where the the Gaussian curvature (the product of the maximum and minimum principal curvatures) of the surface is zero everywhere. Cylinders and cones are examples of a ZGC surface. These surfaces are also called developable surfaces since they can be generated from a piece of paper by rolling and/or bending without cutting. We feel that ZGC surfaces comprise a large and useful class and that they represent a natural step up in complexity from the study of planar surfaces that have dominated previous work in the field. Lines of minimum curvature for a ZGC surface, also called rulings, are straight, *i.e.* it is possible to embed straight lines on a ZGC surface along these rulings.

2.1 Symmetries

We define three types of symmetries, that we call *parallel* symmetry, *line-convergent* symmetry and *skew* symmetry. Parallel and line-convergent symmetries are mainly found in curved surfaces, skew symmetry is usually an indicator of planar surfaces. In our previous work [UN90a] we provided a detailed description of parallel and skew symmetries. Here, we discuss line convergent symmetry in detail.

For curves to be symmetric, certain point-wise correspondences between two curves must exist. We will call the lines joining the corresponding points on the curves as the *lines of symmetry*, the locus of the mid points of these lines as the *axis of symmetry*, and the curves forming the symmetry as the *curves of symmetry*.

2.1.1 Line-Convergent Symmetry

Two image curves C_1 and C_2 are line-convergent symmetric if the tangents of C_1 and C_2, at the corresponding points, intersect along a line, say l, on the image plane. This is shown in figure 2. Parallel symmetry may be thought of as a limiting case of line convergent symmetry where the line of intersection is at infinity. We show later, in section 2.2, that this symmetry is found in curves obtained by cutting a ZGC surface with two non parallel planes. A parallel symmetry also turns into line convergent symmetry under perspective projection [UN91a]. It is also present for limbs of straight homogeneous generalized cylinders [UN90b].

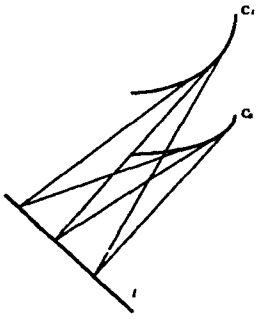


Figure 2: Two line-convergent symmetric curves.

2.2 Symmetries in Surfaces

The symmetries discussed in the previous sections are present in ZGC and planar surfaces. Symmetries also provide strong information about the type of the surface. In [UN90a] we showed that if a closed contour composed of non-limb edges has a skew symmetry, then the contour has to be planar under the assumption of general viewpoint and if the correspondence is static with respect to changing viewpoint. A ZGC surface cut by parallel planes produces parallel symmetry. Moreover, we showed that a figure bounded by one parallel symmetry and one skew symmetry with straight lines of symmetry must be a ZGC surface (assuming general viewpoint in both cases). Line convergent symmetry is produced when a ZGC surface is intersected by non parallel planes.

Theorem 1 *Curves, C_1 and C_2, obtained by cutting a ZGC surface, S, by two non parallel planes, Π_1 and Π_2 project as line-convergent symmetric curves such that the lines joining the corresponding points of the image curves are the projections of the rulings of S and the line l formed on the image plane by joining the intersection points of the tangent lines of the Line-convergent symmetric curves is the projection of the 3-D intersection line of the planes Π_1 and Π_2.*

Proof The above theorem is visualized in figure 3. The key to the proof of this theorem is that the tangent plane, plane T, in figure 3 of the ZGC surface S is same along the rulings of S. Therefore, both the tangent lines, t_1 and t_2, of the curves C_1 and C_2 from points P_1 and P_2 are on plane T. Also the tangent line t_1 is on plane Π_1 and t_2 is on Π_2. Therefore intersection of t_1 and t_2 is necessarily at the intersection point of the three planes Π_1, Π_2 and T. For other rulings the same things repeat for a different T plane, and all the tangent line intersections take place along the line l, the intersection line for planes Π_1 and Π_2. Hence, on the image plane too the intersection of the tangents takes place on the projection of the line l.

Note that the reverse of this theorem, that line-convergent symmetry curves *must* come from non parallel planar cuts of ZGC surfaces, is not valid. However, we believe that it is reasonable to infer that line-convergent symmetry curves are planar cross sections of

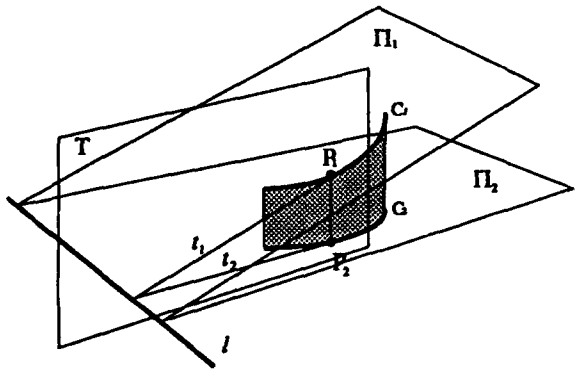


Figure 3: Formation of the line-convergent symmetry with a ZGC surface and two non parallel planes.

ZGC surfaces, if they are terminated by line segments (corresponding to the rulings).

3 Constraints on Surface Shape

We will solve the recovery of shape from contour problem as a constraint minimization problem. The constraints discussed here are the building blocks of constraints and error terms of the minimization. The constraints are originally stated in gradient space. However, the gradient space is not uniform, *i.e.*, a constant shift at the center of the gradient space corresponds to a larger vector difference in 3-D than the same shift somewhere farther away from the center. Therefore, the uniformity of the constraint function implies that the error returned by the function, when not satisfied exactly, depends on the 3-D vector differences rather than the differences in gradient space. A gradient (p,q) corresponds to a 3-D vector of $v=(p,q,1)$. The projection of v on the unit sphere is given by:

$$v_s = \frac{1}{\sqrt{p^2+q^2+1}}(p,q,1) \qquad (1)$$

The vector v_s is the normalized (*i.e.*, $|v_s|=1$) form of the vector v, and v_s is only dependent on the orientation of the vector v, it has no length information. Equation of v_s shows that a constant shift in parameters p and q of v_s has less significance, *i.e.*, affects the components of v_s less, as p and q gets larger. The normalized constraint error functions are formulated to compensate for this behavior of gradient space. The drawback of this normalization is that linear constraint functions are no longer linear.

3.1 Shared Boundary Constraint (SBC)

This constraint relates the orientations of the two surfaces on opposite sides of an edge. The planar version of this constraint has been used since early days in polyhedral scene analysis [Mac73]. Shafer *et al.* [SKK83] first extended it to the case of intersection of curved surfaces. In [UN90a] we also provide a detailed description and derivation of this constraint. Here we just give the normalized version of this constraint, which is a non linear equation. The normalized shared boundary constraint is applied between the gradients (p_1,q_1) and (p_2,q_2) of two surfaces at the point of intersection, and a 2-D vector (x,y), which is the tangent of the curve formed by intersection of the surfaces. The constraint is;

$$SBC(\cdot) = \frac{((p_2-p_1)x+(q_2-q_1)y)^2}{((p_2-p_1)^2+(q_2-q_1)^2+1)(x^2+y^2)} \qquad (2)$$

3.2 Orthogonality Constraint

Certain properties or symmetries invoke the assumption of *orthogonality* in 3-D. The assumption of orthogonality was first studied for skew symmetric planar contours by Kanade [Kan81]. We will assume orthogonality between the axis of parallel symmetry and the lines of parallel symmetry. For a ZGC surface, this is equivalent to slicing the surface along rulings to obtain thin skew symmetric planar strips and assuming that these strips are orthogonally symmetric in 3-D. In [Ulu91] we provide a detailed discussion of this constraint. Here we give the normalized version only. The normalized constraint $Q(\cdot)$ is a function of a surface gradient (p,q) and two image directions (x_1,y_1) and (x_2,y_2) that are hypothesized to be orthogonal in 3-D.

$$O(\cdot) = \frac{((x_1,y_1,px_1+qy_1)\cdot(x_2,y_2,px_2+qy_2))^2}{|(p_1,q_1,1)|^2|(p_2,q_2,1)|^2} \qquad (3)$$

3.3 Equality Constraint

This constraint is applied when two gradients are hypothesized to be equal. Since the gradient space is not uniform, using Euclidean distance between vectors (p_1,q_1) and (p_2,q_2) is not a normalized and uniform error measure. Therefore, square of the sin of the 3-D unit vectors corresponding to the gradients (p_1,q_1) and (p_2,q_2) is used;

$$Eq(p_1,q_1,p_2,q_2) = 1 - \frac{((p_1,q_1,1)\cdot(p_2,q_2,1))^2}{|(p_1,q_1,1)|^2|(p_2,q_2,1)|^2} \qquad (4)$$

4 Recovering ZGCs Cut by Non-Parallel Planes

In order to be able to include the contributions of the constraints from each surface and inter-surface constraints into pool of constraints, appropriate parameter representation for each surface is necessary. The parameterization we use is described below.

4.1 Parameterization of Surfaces

For planar surfaces the gradient space representation (p,q) of the surface normal of the plane is used. This is the natural and most versatile (for our purposes) representation for planar surfaces.

In the following we discuss the parameterization and computation of local surface normals, for the case of ZGC surfaces cut by non parallel planes. As discussed in section 2.2 a ZGC surface cut by parallel planes produces parallel symmetric curves and a ZGC surface cut by non parallel planes produces line convergent symmetric curves. Non parallel cut case is the more general case.

In order to compute line-convergent symmetry between two curves on a general ZGC surface we must try all possible monotonic point correspondences between the curves. This is a very costly search. However, for the case of cylindrical and conic surfaces, the computation of line-convergent symmetry is much simpler. Most of the ZGC surfaces that we encounter in our environment are in fact cylindrical or conic surfaces. Moreover, we can always segment a general ZGC surface into cylindrical and conic sections (mostly at the inflection points) and process each section individually with appropriate constraints applied along the lines of segmentation. For conic surfaces correspondence of line-convergent symmetry is restricted to be along the lines that pass through a common point, the apex of the cone.

To recover the local surface normals for ZGCs cut by non parallel planes, we need to decide which cross section

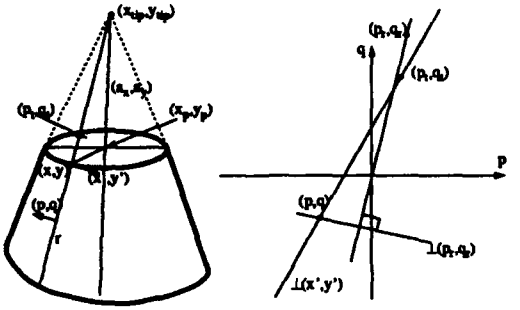


Figure 4: The parameters of a ZGC surface and the constraints in the gradient (p, q) of the surface along the ruling r.

curve is to be made orthogonal to the rulings. This is because the cross section planes are not parallel to each other.

A ZGC surface cut by non parallel planes have five degrees of freedom. Of the five degrees of freedom, the four parameters are the gradients (p_t, q_t) and (p_b, q_b) of the top and the bottom planes cutting the ZGC surface, and the u parameter discussed below.

We can model a conic surface by using any 3-D axis that goes through the apex of the cone. We use the 3-D axis that projects as the 2-D axis of the straight edges of the cone. If the image direction of the axis is (a_x, a_y) then the 3-D direction of the axis in gradient space representation is (ua_x, ua_y) where u is a free variable, and it is the fifth parameter of the ZGC representation.

4.2 Recovering ZGCs Cut by Two Planes

Consider figure 4, let (p_t, q_t) be the gradient of the cross section plane that is chosen to be made orthogonal to the surface. The gradient (p, q) of the surface along the ruling r is given by combination of two linear constraints given below. The first one is the shared boundary constraint, given in section 3.1, between gradients (p, q) and (p_t, q_t) using the tangent (x', y') of the intersection curve at the point the curve touches the ruling r. The equation of the constraint that $(p - p_t, q - q_t) \cdot (x', y') = 0$ This constraint is shown in the gradient space by the line labelled $\perp (x', y')$ in figure 4.

The second constraint is that the 3-D gradient $(p_r, q_r, 1)$ of the ruling r must must be orthogonal to the 3-D gradient, $(p, q, 1)$, of the surface along ruling r, that is; $(p, q, 1) \cdot (p_r, q_r, 1) = 0$ In figure 4 this constraint is shown by the line labelled $\perp (p_r, q_r)$, which is the orthogonal line of the gradient (p_r, q_r), *i.e.*, the gradient of the set of the directions that are orthogonal to (p_r, q_r) in 3-D. Note that this line is also orthogonal to the 2-D direction of the image of the ruling.

The gradient (p_r, q_r) of the ruling r is obtained by reconstructing the axis line (a_x, a_y), and the line between points (x, y) and (x_p, y_p) in 3-D (*i.e.*, computing the z coordinates of these image points). Since the gradient of the axis line is (uax, uay), fixing $z_p = 0$, the z coordinate of the point (x_{tip}, y_{tip}), z_{tip}, is given by;

$$z_{tip} = \frac{x_{tip} - x_p}{ua_x} \tag{5}$$

The z coordinate of the point (x, y) is computed using the gradient (p_t, q_t) as;

$$z = p_t(x - x_p) + q_t(y - y_p) \tag{6}$$

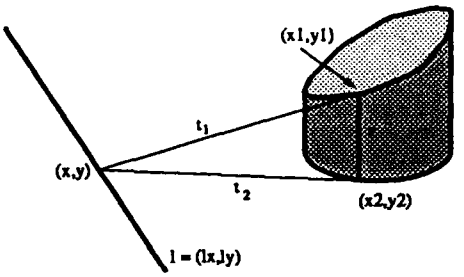


Figure 5: Constraints on the orientation of the cutting planes of a ZGC surface.

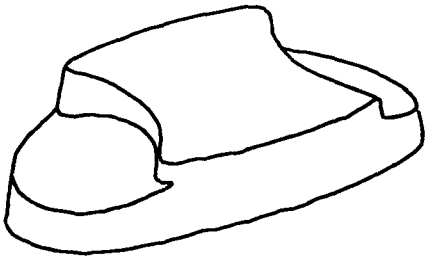


Figure 6: The telephone example.

Then the gradient (p_r, q_r) of the ruling is given by;

$$(p_r, q_r) = \frac{1}{z_{tip} - z}(x_{tip} - x, y_{tip} - y) \tag{7}$$

These formulas are exactly the same for ZGCs cut by parallel planes. In that case the result is independent of which parallel symmetry curve used.

Compared to ZGCs cut by parallel planes, for a ZGC surface cut by non parallel planes there are two additional unknowns, which are the gradient parameters of the second cutting plane. Also, there are two additional constraints on the orientation of the planes cutting the ZGC surface. These constraints are not needed compute the local surface normals of a ZGC surface. In fact the gradient of the second plane (*i.e.*, the plane that is not chosen to be made orthogonal to the surface) is not needed at all for that purpose. However, for the multiple surface recovery algorithm the gradient of the second plane is needed so are these constraints. Here we state the constraints, they will be used in section 5.1.

The first one is a shared boundary constraint; for a ZGC surface S, with line-convergent symmetry, let (p_t, q_t) be the gradient of the top plane and let (p_b, q_b) be the gradient of the bottom plane, and let the intersection line have direction (l_x, l_y) on the image plane. Since the top and the bottom planes actually intersect each other along the line l in 3-D we have the shared boundary constraint as;

$$SBC(p_t, q_t, p_b, q_b, l_x, l_y) = 0 \tag{8}$$

Consider figure 5; let (p_r, q_r) be the local surface gradient of the surface S along ruling r. This constraint enforces that the 3-D lines t_1 and t_2 be on the same tangent plane having gradient (p_r, q_r). The constraint is:

$$t_1 \times r \equiv t_2 \times r \tag{9}$$

5 Combined Shape Recovery

Many objects of interest consists of several curved surfaces. Here the recovered 3-D individual surfaces must

be in agreement with the neighboring surfaces, *i.e.*, surfaces sharing a common boundary. We describe a technique for such integrated multiple surface recovery for objects consisting of planar and ZGC surfaces. Figures 1 and 6 show some sample objects.

The shape of all the surfaces is recovered simultaneously by finding appropriate values for the parameters of each surface. The values of the surface parameters are computed by solving the following constraint minimization problem;

$$minE_i \quad \text{subject to} \quad E_x = 0 \tag{10}$$

where E_i stands for error terms resulting from internal constraints of each surface and E_x are the external terms, that is, the constraints obtained by intersection of surfaces.

5.1 Internal Constraints

The internal constraints are the constraints obtained from the regularity assumptions of each surface. In general they have the following form:

$$E_i = \sum w_p E_p + \sum w_o E_o + \sum w_c E_c \tag{11}$$

Where each ws are weight and E_p is the error term for the orthogonality constraint of the planes, E_o is the error term for the orthogonality constraint of the ZGC surfaces and E_c is the error term for the implicit constraint of the parameters of ZGC surfaces. These error terms are described in more detail in the following.

E_p is the error term for the orthogonality constraint of planes. If a planar surface has a skew symmetry then this is the orthogonality function of the lines and axis of skew symmetry as given in 3, where (x_1, y_1) and (x_2, y_2) are the image directions of the lines of symmetry and the axis of symmetry and (p, q) is the gradient of the plane. w_p is the weight of E_p and is proportional to the total length of the contour enclosing the surface. The formula used for w_p is $w_p = \sqrt{l_c}$ where l_c is the total length of the curve enclosing the surface. If the surface does not have skew symmetry E_p is zero.

E_o is the error term for the orthogonality of ZGC surfaces. As in our previous work on ZGC surfaces [UN90a], we choose to make the directions of the rulings orthogonal to the tangents of the parallel or chosen line convergent symmetry curve in 3-D.

$$E_o = \frac{1}{N} \sum_i O(p_i, q_i, x'_i, y'_i, x_{tip} - x_i, y_{tip} - y_i) \tag{12}$$

here $O(\cdot)$ is the orthogonality constraint given in 3 and for $i \in [0, N-1]$, at i^{th} location; (p_i, q_i) is the local surface normal represented in gradient space (x'_i, y'_i) is the tangent of the line convergent symmetry curve that is chosen to be orthogonal to rulings, (x_i, y_i) is the location of the point that the ruling meets with the line convergent symmetry curve, and (x_{tip}, y_{tip}) is the location of the apex of the cone. These are donated as (p, q), (x', y'), (x, y), $(x_{tip}, ytip)$ respectively in figure 4. However computing Ξ requires computation of local normals at each point on the surface at each iteration of minimization, which is very time consuming. Also, since it is a huge non linear expression, it creates many local minima around the true minima, which the minimization routine gets stuck. Therefore from our experiments we use an approximation of E_o, $\hat{E}_o$, such that $\hat{E}_o$ is quadratic and error given by $\hat{E}_o$ is very similar to the one given by E_o. $\hat{E}_o$ has the following form:

$$\hat{E}_o = E_{ax} + E_t \tag{13}$$

where $E_{ax} = \cos^2(\alpha)$, and α is the angle between the gradient (p_t, q_t) of the plane containing the parallel (or line-convergent) symmetry curve that is decided to be made orthogonal and the direction of the image axis (a_x, a_y) of figure 4. Motivation for E_{ax} is based on the observation that minimum of E_o in general occurs when (p_t, q_t) of figure 4 is along a line in $p-q$ space that passes through the origin and is parallel to the image axis. $E_t = (u_{int} - u)^2$ where u is the u-parameter of the ZGC surface and u_{int} is set at the initialization by minimizing orthogonality error E_o given in equation 12. In our implementation we tried both E_o and $\hat{E}_o$. It turns out that $\hat{E}_o$ performs better (in the sense of stability) because it is a simpler error function and creates fever local minima. On can use $\hat{E}_o$ as an initializer for E_o, but in our experiments another run of minimization with E_o over $\hat{E}_o$ was not necessary. w_o is the weight of the orthogonality term and is proportional to the total length of the perimeter of the surface, $w_o = \sqrt{l_c}$ where l_c is the total length of the contour enclosing the surface.

E_c is the error term for implicit constraints of the parameters of ZGC surfaces. Let (p_t, q_t) and (p_b, q_b) be the gradients of the planes containing the two parallel (or line-convergent) symmetry curves of the ZGC surface. If the ZGC surface has a parallel symmetry then (p_t, q_t) should be equal to (p_b, q_b), therefore, $E_c = Eq(p_t, q_t, p_b, q_b)$, where $Eq()$ is given in equation 4. If the ZGC surface has an line-convergent symmetry then E_c is the addition of the constraints given in equations 8 and 9. w_c is the weight and is inversely proportional to the eccentricity of the parallel (or line-convergent) symmetry curves. If the parallel (or line-convergent) symmetry curves are highly eccentric, *i.e.*, they are almost straight, then the weight of this constraint is low. The formula for $w_c = 1/ecc$, were ecc is the eccentricity of the total cross section curve (the eccentricity of a curve is computed by using the scattering matrix of the curve).

5.2 External Constraints

External constraints are the inter-surface restrictions imposed by each surface on neighboring surfaces. Extremal constraints have the following form:

$$E_x = \sum wE_{pp} + \sum wE_{pz} + \sum wE_{zz} \tag{14}$$

where w is the weight of each constraint and is equal to $\sqrt{l_c}$ where l_c is the length of the curve produced by intersection of the surfaces. E_{pp}, E_{pz} and E_{zz} are the error terms for shared boundary constraint between planes, between planes and ZGC surfaces, and between ZGC surfaces respectively. In detail, the individual error terms are:

E_{pp} is the error of the shared boundary constraint between the gradients of the two intersecting planes as given in equation 2.

E_{pz} is the error term for shared boundary constraint between a plane and a ZGC surface. There are two possibilities; the intersection is along a ruling of the ZGC

surface or the intersection is along a parallel (or line-convergent) symmetry of the ZGC surface. If the intersection is along the ruling of the ZGC surface then E_{pz} is the shared boundary constraint as given in equation 2 between the gradient of the plane and the local surface normal of the ZGC surface at the ruling of intersection. If the intersection is along one of the parallel (or line-convergent) symmetry curves then;

$$E_{pz} = Eq(p, q, p_t, q_t) \quad (15)$$

where (p_t, q_t) is the parameters of the ZGC surfaces which is the gradient of the plane containing the intersection curve and (p, q) is the gradient of the planar surface.

E_{zz} is the error term for the shared boundary constraint between two ZGC surfaces. There are various ways two ZGC surfaces may intersect each other. Here we only handle the intersections that produce a planar intersection curve. There are two types of such intersections: along the rulings of the ZGC surfaces or along the parallel (or line-convergent) symmetry of the ZGC surfaces. If the intersection is along the rulings of the ZGC surfaces then Shared Boundary Constraint given in equation 2 is applied between the local surface normals of ZGC surfaces at the ruling of intersection. If the intersection is along the parallel (or line-convergent) symmetry curves, then, let (p_1, q_1) and (p_2, q_2) be the gradients of the planes containing the intersection curve in the representations of the first and the second intersecting ZGC surfaces. The error term is:

$$E_{zz} = Eq(p_1, q_1, p_2, q_2) \quad (16)$$

When two ZGC surfaces intersect each other along their parallel symmetry (or line-convergent symmetry) curves, how orthogonal both surfaces can be made depends on how parallel their image axes are. Therefore we form a new orthogonality error term E_{on} for the intersecting ZGC surfaces to replace their original orthogonality error terms (E_o's). Let α be the angle between the image axis of these surfaces, let E_{o1} and E_{o2} be the error terms for the orthogonality of the intersecting ZGC surfaces. Then the new combined orthogonality error term is;

$$E_{on} = \cos^2(\alpha)(E_{o1} + E_{o2}) + \sin^2(\alpha)(E_{o1}E_{o2}) \quad (17)$$

E_{on} emphasizes the orthogonality of both of the ZGC surfaces when the image axis are almost parallel to each other, and it emphasizes the orthogonality of either of the ZGC surfaces when the image axes are almost orthogonal to each other.

5.3 Solving Constraint Equations

The total error function E is solved using a constraint minimization technique, where E_x consists of "must-satisfy" external constraints and E_i consists of assumption driven error terms as defined earlier. To solve this constraint minimization (where the constraints are non-linear), the problem is converted into a minimization form as follows:

$$\lim_{\lambda \to inf} minE = \lim_{\lambda \to inf} min(E_i + \lambda E_x) \quad (18)$$

That is, E is minimized for successively larger values of λ, thus, emphasizing E_x more at each minimization cycle. At the end E_x constraints are satisfied almost exactly and E_i's are minimized to the extent possible. In our implementation we increased λ from 1 to 100 in exponential steps of 3.5 (that is λ =1, 3.5, 12.25, ... etc.).

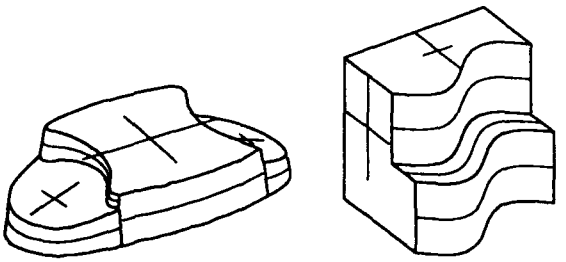


Figure 7: The segmented surfaces, and the symmetries computed for each surface. The skew symmetry of planar surfaces are shown by crosses, the long line is the axis of symmetry and the short one is the direction of the lines of symmetries. Parallel and line-convergent symmetries are shown by their curved axis only.

For the minimization, a gradient descent algorithm is used. The set of parameters of the surfaces ((p, q)'s and u's) minimizing E is taken as the solution set and used to reconstruct the local surface gradients.

Initial values of the parameters of E, *i.e.*, the parameters of all the surfaces involved in E, are computed by an initializer. The initializer starts with an arbitrary ZGC surface, and sets its parameters as if it is an isolated surface. Then, the initializer sets the parameters of the neighboring surfaces by keeping them consistent with the first surface, and, the neighbors of these surfaces are processed progressively until the parameters of all the surfaces are initialized.

6 Implementation and Results

For the results shown in this section the following implementation is used. The input to the program are segmented curves represented as a list of points that define the contour of each object. However we do not assume that the input curves are noise free. These segmented curves are grouped into closed regions using continuity. Each closed region is taken to correspond to an object surface. Next, we find symmetries among segments of a surface. The details of symmetry finding is presented in [Ulu91].

The surfaces containing parallel or line convergent symmetric segment pair are treated as curved and others are treated as planar. For curved surfaces the curves joining parallel symmetric curves are checked if they are straight to confirm that the surface is a ZGC.

Some surfaces are combination of various curved surfaces and there is no distinctive boundary between them. This is the case for the curved surfaces of the object in figure 6. Such surfaces contain more than one parallel (or line-convergent) symmetry and they are segmented into smaller surfaces containing only one parallel (or line-convergent) symmetry. Figure 7 shows the segmented surfaces and the symmetries (skew, parallel or line-convergent) for each surface.

The constraints for each surface and inter-surface constraints including the ones for the newly formed intersections are extracted forming the error function E. Then E is minimized by the constraint minimization technique discussed in section 5.3.

Figure 8 shows the final results for the objects in figures 1 (a) and 6. This process takes approximately 2 minutes for an object on a Symbolics 3645 computer running in LISP. The computed surface normals are shown by needle diagrams, as needles sticking to the surface in the direction of the local surface normals. For planar surfaces, a small coordinate frame, with a triangle

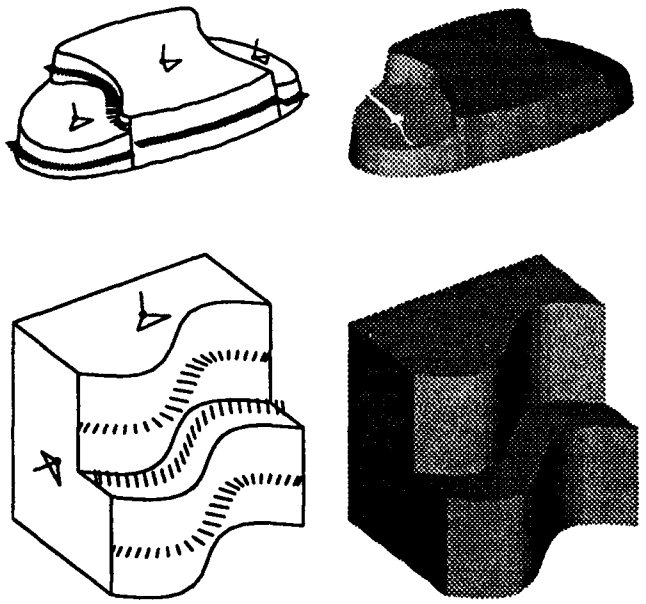


Figure 8: The needle and the shaded images obtained from the computed surface normals for the objects in 6.

at the base, is used to better show the computed surface normal. We also provide the shaded images of the objects computed by using the surface normals, a Lambertian reflectance model and a point light source. Note that, for the bottom object in figure 8, the middle surface is initially classified as a curved surface due to the parallel symmetry it has. However, its perceived as a planar surface effected by the planarity of the top surface as discussed in the Introduction. The final result of the minimization, in fact, shows the middle surface as planar within error bounds of the minimization. We believe that the computational results are in agreement with human perception, though we have not attempted a quantitative comparison.

7 Conclusion

We have described a technique for recovering 3-D shape of objects consisting of zero-Gaussian curvature (and planar) surfaces. Our method incorporates the constraints imposed by all the surfaces simultaneously. As shown by an example, this can result in fixing the shapes of some surfaces which are ambiguous otherwise.

We have attempted to show the accuracy of our results by comparison with human perception. For shape from contour analysis, the only *ground truth* is really in human perception, for even if the given contour was obtained by a real object, it *could* have been also obtained by any number of other objects as well. Our system has a certain notion of preference that is based partly on geometrical analysis and partly on perceptual observations.

The current system assumes that clean and complete boundaries are given. This can not be expected to be the case for real objects in complex environments. To cope with these difficulties will require incorporation of some perceptual organization techniques. We believe that our methodology will help in this step too, as we are able to provide strong constraints that hypotheses of a perceptual organization system can be tested against. This is the topic of separate, current research in our laboratory.

References

[BT81] H.G. Barrow and J.M. Tenenbaum. Interpreting line drawings as three dimensional surfaces. *Artificial Intelligence*, 17:75–116, 1981.

[BY84] M. Brady and A. Yuille. An extremum principle for shape from contour. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:288–301, 1984.

[Clo71] M.B. Clowes. On seeing things. *Artificial Intelligence*, 2(1):79–116, 1971.

[HB88] R. Horaud and M. Brady. On the geometric interpretation of image contours. *Artificial Intelligence*, 37:333–353, 1988.

[Huf71] D.A. Huffman. Impossible objects as nonsense sentences. *Machine Intelligence*, 6:295–323, 1971.

[Kan81] T. Kanade. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17:409–460, 1981.

[KK83] T. Kanade and J. R. Kender. Mapping image properties into shape constraints: Skew symmetry, affine transformable patterns, and the shape from texture paradigm. In J. Beck, B. Hope, and A. Rosenfeld, editors, *Human and Machine Vision*, pages 237–257. Academic Press, 1983.

[Mac73] A.K. Mackworth. Interpreting pictures of polyhedral scenes. *Artificial Intelligence*, 4:121–137, 1973.

[Nal87] V. Nalwa. Line drawing interpretation: Bilateral symmetry. In *Proceedings of the Image Understanding Workshop*, pages 956–967, 1987. Los Angeles.

[PCM89] J. Ponce, D. Chelberg, and W. B. Mann. Invariant properties of straight homogeneous generalized cylinders and their contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(9):951–966, 1989.

[SKK83] S. A. Shafer, T. Kanade, and J.R. Kender. Gradient space under orthography and perspective. *Computer Vision, Graphics and Image Processing*, 24:182–199, 1983.

[Ste81] K. A. Stevens. The visual interpretations of surface contours. *Artificial Intelligence*, 17:47–73, 1981.

[Ulu91] F. Ulupinar. *Perception of 3-D Shape from 2-D Images of Contours.* PhD thesis, University of Southern California, 1991.

[UN90a] F. Ulupınar and R. Nevatia. Perception of 3-d surfaces from 2-d contours. In *Proceedings of the 10th International Conference on Pattern Recognition*, pages 147–154, 1990. Atlantic City, New Jersey.

[UN90b] F. Ulupınar and R. Nevatia. Shape from contour: Shgcs. In *Proceedings of the 3rd ICCV*, 1990. Japan.

[UN91a] F. Ulupinar and R. Nevatia. Constraint for interpretation of line drawings under perspective projection. *Computer Vision, Graphics and Image Processing*, 52:674–676, 1991.

[UN91b] F. Ulupınar and R. Nevatia. Recovering shape from contour for constant cross section generalized cylinders. In *Proceedings of Computer Vision and Pattern Recognition Conference*, 1991. Maui,Hawai.

[Wei88] I. Weiss. 3-d shape representation by contours. *Computer Vision, Graphics and Image Processing*, 41:80–100, 1988.

[XT87] G. Xu and S. Tsuji. Inferring surfaces from boundaries. In *Proceedings of the 1st ICCV*, pages 716–720, 1987. London.

Recovering 3-D Wire Frames From Line Drawings

Martin A. Fischler and Yvan G. Leclerc[1]

Artificial Intelligence Center
SRI International
333 Ravenswood Ave., Menlo Park, CA 94025
(fischler@ai.sri.com leclerc@ai.sri.com)

Abstract

Line drawings provide an effective means of communication about the geometry of 3-D objects. An understanding of how to duplicate the way humans interpret line drawings is extremely important in enabling man-machine communication with respect to images, diagrams, and spatial constructs. In particular, such an understanding could be used to provide the human with the capability to create a line-drawing sketch of a polyhedral object which the machine can automatically convert into the intended 3-D model.

A recently published paper (Marill 1991) presented a simple optimization procedure supposedly able to duplicate human judgement in recovering the 3-D "wire frame" geometry of objects depicted in line drawings. Marill provided some impressive examples, but no theoretical justification for his approach. In this paper we introduce our own work by first critically examining Marill's algorithm. We provide an explanation for why Marill's algorithm was able to perform as well as it did on the examples he presented, discuss its weaknesses, and show very simple examples where it fails. We then provide an algorithm that improves on Marill's results. In particular, we show that an effective objective function must favor both symmetry and planarity—Marill deals only with the symmetry issue. By modifying Marill's objective function to explicitly favor planar-faceted solutions, and by using a more competent optimization technique, we were able to demonstrate significantly improved performance in all of the examples Marill provided and those additional ones we constructed ourselves. Finally, we examine some questions relevant to the implications of this work for understanding the human ability to interpret line drawings.

[1] The work reported here was partially supported by the Defense Advanced Research Projects Agency. We gratefully acknowledge the valuable discussions with Aaron Bobick and Tom Strat regarding both the content and organization of this paper.

1 Introduction

The interpretation of line drawings has been an important focus for research in machine vision since the field's inception. There seems to be little question that human subjects can easily recover 3-D models from the 2-D line drawings depicting many classes of objects. One such class of special interest has been called the "blocks world." This class consists primarily of polyhedral solids in three dimensional Euclidean space and the projections of the visible edges of these objects onto a 2-D plane (which we call the line drawing). Given a single line drawing of a blocks world scene, normal human subjects will usually arrive at the same 3-D interpretation, even though there may be a very large number of possible 3-D objects that could have produced the given drawing.

Beginning with the the work of Guzman in 1968, there has been a concerted effort by vision researchers to develop an algorithmic procedure that could duplicate human performance in interpreting line drawings, at least with respect to blocks world objects. A significant body of work in this area was produced by such prominent scientists as Clowes (1971), Huffman (1971), Waltz (1972), Mackworth (1973), Kanade (1980), Draper (1981), and Sugihara (1982, 1984). However, the problem as originally formulated, devising a procedure for recovering psychologically plausible 3-D models from line drawings, remains unsolved.

The earliest work by Guzman was heuristic in nature, failed in many cases where humans had no trouble in finding appropriate interpretations, and did not actually return a 3-D model, but rather partitioned the scene into separate polyhedral objects. Clowes, Huffman, Waltz, Mackworth, and Kanade formalized and extended the work of Guzman, but did not solve the original problem. They were (usually) able to label the edges of the line drawing to correctly reflect a consistent 3-D interpretation if one existed, or could assert that the drawing did not correspond to a realizable blocks world scene. Mackworth and Kanade explicitly exploited the planarity of the faces of blocks world and "Origami" objects (by employing a "gradient space" representation) to accomplish

a form of semi-quantitative recovery. In addition to consistent edge labeling, they could also constrain the relative orientation of the faces of the target 3-D model. The labels could describe the edges as being convex, concave, occluding edges, etc., but still, for the general case, no explicit 3-D model was returned (without introducing additional constraints) and the algorithms would make occasional errors.[2]

In a series of papers, Sugihara reformulated the realizability and recovery problems for line drawings of polyhedra (both with and without hidden line removed) in purely algebraic terms. He required as input a specification of the vertices defining each of the individual planar faces of the polyhedra, and also required that the implied line drawing be a general-position projection of the polyhedra. With this approach he succeeded in providing an algebraic criterion as a necessary and sufficient condition for a line drawing to represent a physically realizable polyhedral object. He could also constrain the space of feasible solutions, and obtain a unique solution if enough additional constraints were provided. These additional constraints were obtained from information beyond that provided by the line drawing (e.g., shading or texture information). Sugihara's work was an important advance, but again, it fell short of the original goal. It will rarely be the case that a unique reconstruction is implied by the line drawing, and thus, the primary objective of duplicating human performance in this regard is not met.[3]

Our motivation for writing this paper was supplied, in part, by a recent publication authored by T. Marill (1991). He refocused on the original problem of human interpretation of single line drawings as three-dimensional structures; he did not restrict his universe to blocks world objects nor did he demand that the line drawings be complete. The surprising thing about his work was that he used an optimization approach involving (seemingly) an almost trivial objective function, and the simplest possible gradient descent algorithm to find a solution, and yet, provided examples of reconstructed objects that were, intuitively, extremely good. (Figure 1, Examples A through I, shows the line drawings used in Marill's experiments.) However, his paper provided no justification for why the algorithm should work, and thus no basis for judging its generality or insight into how it could be improved (should this be desirable).

In this paper we introduce our own work by first critically examining Marill's algorithm. We provide an explanation for why Marill's algorithm was able to perform as well as it did on the examples he presented, discuss its weaknesses, and show very simple examples where it fails (Figure 1, Examples J through N). We then provide an algorithm that improves on Marill's results for all nine of his examples, and also successfully deals with the simple cases where Marill fails. Finally, we examine some questions relevant to the implications of this work for understanding the human ability to interpret line drawings.

We see the work described in this paper as being of both theoretical and practical interest. The practical utility of this work is its relevance to man-machine communication about 3-D structures via line drawings - in particular, providing the human with the capability to create a line drawing sketch of a polyhedral object which the machine can automatically convert into the intended 3-D model. Deficiencies in providing a complete theory are not fatal, since auxiliary information can always be supplied interactively to resolve ambiguities, but the underlying theory should reduce this "side communication" to a minimum.

2 Marill's Algorithm

Marill's algorithm consists of two components, an objective function and a gradient descent optimization procedure for finding a local minimum of this objective function. The objective function is simply the standard deviation of all of the angles (SDA) in the recovered 3-D object with respect to their common mean. Marill calls the minimization of the SDA the MSDA principle.

The input line drawing is specified as a set of points (vertices) and lines; each point is represented by an (x, y) coordinate pair, and each line is represented by an integer pair corresponding to the sequence numbers of the two points it joins. The representation of the recovered 3-D object involves supplying a third (z) coordinate for each of the originally specified points. This is called the orthographic extension of the line drawing, and is actually a wire frame rather than a solid object.

To evaluate the objective function for a given proposed solution, every pair of lines terminating on a point (as defined in the input specification) is considered to form a separate angle. Thus, if five lines terminate on the same point, every potential 3-D solution contains ten angles at this point which contribute to the objective function. Note that the intersection, between two lines that happen to cross at *intermediate* points of their extent in the line drawing, is not treated as a vertex, and does not contribute to the objective function (even if the lines lie in the same plane in the 3-D reconstruction). Similarly,

[2]Gradient space, originally conceived of by James Clerk Maxwell in 1864 (see (Whitely 1986)) and rediscovered by D. A. Huffman, provides only necessary conditions for planar realizability of general polyhedral objects with hidden lines removed, and thus consistent edge labeling is possible for impossible blocks world and Origami objects. Further, the labeling/recovery algorithms were not always competent to find an existing solution.

[3]There were some other problems of lesser significance for our purposes in this paper. For example, the algebraic formulation was sensitive to computation round-off errors, and digitization errors in specifying the line drawing; a realizable object could be rejected because of such minor numeric inaccuracies. Sugihara dealt with this problem by adding an optimization step to his algorithm which could find a feasible reconstruction if the input drawing was an almost correct specification.

two *distinct* vertices can have the same (x, y) coordinates in the line drawing, but then the line segments terminating on the distinct vertices do not interact to form angles (even if the vertices coincide in the 3-D reconstruction).

Thus, given a line drawing with n vertices, each possible orthographic extension is represented as a z-vector having n components; the corresponding angles and SDA are computed to evaluate the proposed solution. Marill uses a gradient-descent technique to search for a best answer, recognizing that this is simply a heuristic and that this approach will only find a single local minimum of his objective function. The input object has all of its z values initially set to zero; i.e., it is a flat object lying in the (x, y) plane. At each stage in the search, the SDA of the current z-vector is computed and the program then looks at the children of the current vector. These $2n$ children are all of the vectors one step size away from the current vector, and are formed by both adding and subtracting a specified value (Δz) to each of the n components in the current z-vector. The value of the SDA is computed for each of these $2n$ children, and the child with the minimum SDA is selected as the new current vector. This process is repeated until no improvement in the SDA is obtained, and the resulting z-vector is returned as the solution for the first of three rounds of gradient descent. Each additional round uses a smaller Δz and begins with the result of the preceding round. Marill experimentally found effective values of Δz for his three rounds to be 1, 0.5, and 0.1.

Figure 2 shows a line drawing, it's internal representation as described above, and the reconstructions using Marill's algorithm and the algorithm we describe in Section 3.

In the top left window of the figure is the input line drawing (with the vertices numbered for reference by the written representations below). The four windows on the top right show two views of Marill's reconstruction and two views of our reconstruction. In the middle of the figure is a table showing the internal representation of the input line drawing. In the first row are the (x, y) coordinates of the vertices, in the order shown on the line drawing.[4] In the second row are the integer pairs representing the lines in the drawing. In the third row are the sequences of vertices corresponding to the planar faces derived according to the rules of Appendix A (see Section 3). The reconstructions will be discussed in detail in Section 3.1.

2.1 Marill's Examples

Marill described the application of his algorithm to Examples A through I of Figure 1. We categorize these examples along the following dimensions (based on the appearance of the input drawing and on the characteristics of the recovered 3-D object):

a) • Three Dimensional [A B D E F G H I]
   • Flat [C]

b) • Blocks World (planar faced solids with occluded edges not rendered) [B H I]
   • Origami (planar faced, possibly hollow) [C F]
   • Wire Frame of Blocks World Object (all edges of a blocks world object are given, and additional lines between vertices of a planar face may be added) [A D G]
   • Restricted Wire Frame (every closed circuit of lines, without interior lines in the given input representation, corresponds to a planar face) [E]
   • Non-Planar Wire Frame (none of the above)

c) • Symmetric [A B C E G H]
   • Asymmetric [D F I]

d) • All Angles (approximately) Equal [A B E F H]
   • A Few Distinct but Mostly Repeated Angles [C G I]
   • Mostly Unequal Angles [D]

For the purposes of our discussion, we use Marill's categorization and augment it with our own subjective evaluation where we disagree or need to add additional attributes to those Marill provides. It is important to remember that Marill always returns a wire frame as his solution, regardless of the categorization of the object. Thus, we would call the wire frame of a blocks world object a correct solution if it was a geometrically correct representation of the 3-D geometry of the edges of the psychologically plausible blocks world object whose orthographic projection corresponded to the input line drawing, even though the wire frame does not provide an explicit representation of the grouping of lines into faces, etc.

Examples A, B, E, F, and H can all be visualized as approximately equiangular three dimensional objects. That is, each of the objects has an equiangular 3-D wire frame as a psychologically plausible solution. Since these equiangular solutions exactly satisfy Marill's minimum standard deviation of angles (MSDA) criterion, it is obvious why Marill's objective function should prefer what we accept as the correct solutions in these cases. In the other four cases, supposedly representative examples of the ability of Marill's algorithm to deal with complicated structures having unequal angles, reasonably correct solutions are also recovered, and it is this performance we wish to understand.

2.2 The Performance of the MSDA Principle

Given its overall simplicity, it would be quite remarkable if the MSDA principle generally converged to a psycho-

[4] For simplicity, the vertices are written using only two digits of precision in the table. However, we used the full 32-bit precision of the projection in the internal representation used by the algorithms.

logically correct reconstruction. (A psychologically correct reconstruction of a line drawing is the one that virtually all people make.) Unfortunately, it is rather easy to find examples where this is not the case, contrary to Marill's implied competence for the principle.

Examples J through N of Figure 1 are line drawings for which Marill's algorithm converged to solutions that are clearly psychologically incorrect, even though these drawings are not significantly more complicated or more asymmetric than the examples that Marill used (Figures 2, 3, 4, 5, and 6 illustrate both Marill's reconstructions and our reconstructions, as described in Section 3.) In Examples J, K, and M it would appear that the fault could lie with Marill's overly simple gradient descent algorithm because the SDA of the psychologically correct answer is lower than the SDA for the solution Marill actually obtains. Thus, one can argue that a more competent global search strategy could have found the psychologically correct answer using the same objective function. However, Examples L and N are line drawings for which the SDA of Marill's solution is significantly lower than that of the psychologically correct solution. Thus, the MSDA principle is clearly not adequate to *reliably* handle even simple line drawings.

Before discussing ways of augmenting the MSDA principle to obtain a more competent algorithm, we attempt to explain the performance of MSDA for line drawings depicting objects that are not equiangular.

2.3 Evaluating the Performance of the MSDA Principle

It is not immediately obvious why the MSDA principle should prefer a psychologically plausible answer if the object depicted in the line drawing contains two or more significantly different angles (e.g., C, D, G, I, and J). Marill offers no explanation for this phenomena, and thus no way to judge the conditions under which his algorithm should be expected to succeed or fail. In this section we provide a partial explanation for cases (such as C, G, J, K, and L) that have critically important attributes—the psychologically correct reconstruction is a 3-D planar-faced object whose faces are either equiangular or form "complete-star" configurations (see Appendix B).

To establish the role played by the above geometric attributes, we define the *planar orthographic extension of a simple closed 2-D circuit* in a line drawing to be any orthographic extension for which the corresponding 3-D contour is planar. If a line drawing contains more than one simple closed 2-D circuit, then a *planar orthographic extension of the entire line drawing* exists if we can cover the line drawing with a set of simple closed 2-D circuits such that (a) every angle in the drawing is included in at least one circuit, and (b) each circuit projects to a 3-D planar contour.[5]

In Appendices B, C, and D, we provide a number of theorems that are pertinent to understanding the effectiveness of the MSDA principle applied to planar orthographic extensions. The main theorem, Appendix D, asserts that solutions with certain symmetries correspond to the global minimum of the SDA over all planar orthographic extensions (the specific symmetry condition we examine is that all faces must either be equiangular or form complete-star configurations).

Consequently, if there were some way to consider as possible solutions only the *planar* orthographic extensions of a line drawing, (such as the psychologically plausible solutions for Examples A, B, C, G, J, K, and L), these solutions will be global minima of the SDA because of the angular symmetry they exhibit. We sho in Example L that Marill's algorithm is not constraine to search only for planar solutions; while it will also find solutions with non-planar facets that have lower SDAs than the planar solutions, there is still the possibility that MSDA shows at least a weak inherent preference for planarity. While we cannot completely rule out this possibility, it appears that the geometric constraints inherent in the specific examples Marill selected, rather than MSDA itself, are largely responsible for finding planar faceted solutions. Specifically, triangles in the line drawing will always produce planar facets in the orthographic extension, and as we prove in Appendix B, a closed four sided polygonal space curve with 90 degree angles at each vertex will always be a planar configuration. Since in Marill's examples listed above, all the facets satisfy these two geometric conditions, we see why both the desired planarity and symmetry are present in the computed solutions.[6]

Marill offers only two examples (D and I) which are not clear instances of the above analysis (all angles equal, or symmetric planar facets). His solution for Example I is at least questionable if not incorrect. However, this solution has almost all of its angles equal to 90 degrees and so it needs no further explanation if we accept it as correct.

Marill's solution to the asymmetric drawing of Example D looks very reasonable; it has all its angles fairly well distributed between 40 and 70 degrees and it is not possible to find a more symmetric (equiangular) orthographic extension for this line drawing. However, be-

[5] We note that while there generally can be many different ways of covering a line drawing, those of blocks-world objects with hidden lines removed will be covered uniquely if we demand that the interior of the 2-D circuits be free of any lines. We also note that it is not always possible to cover a line drawing with *simple closed* circuits corresponding to the specified planar facets of a given orthographic extension (see Example N). It may also be the case that a given covering has no non-trivial orthographic extension with planar facets as specified, as in Example O.

[6] There is one facet, in Example H, that is an exception to this statement. However, there are enough other geometric constraints in this one case to enforce planarity.

cause the input line drawing is a completely connected set of triangular faces, every possible orthographic extension will have planar facets and an acceptable topology; it is hard to produce an unacceptable reconstruction for this drawing.

In summary, there is an understandable reason why Marill's MSDA principle will sometimes tend to select planar symmetric 3-D wire frames when a purely equiangular solution is not possible. But we also see that MSDA will make unacceptable errors, even in simple cases, because it is not constrained to prefer solutions with planar facets unless the geometry of the line drawing itself forces planarity.

3 An Augmented MSDA Algorithm

What's missing in Marill's MSDA principle is a means for enforcing the planarity of specified facets. In extending the MSDA principle, we introduce an additional component to the objective function, called DP, that increases in value as the facets deviate from planarity. The new objective function, $E(\lambda)$, is the sum of the previously defined SDA term and DP:

$$E(\lambda) = \lambda SDA + (1 - \lambda)DP.$$

Thus, minimizing $E(\lambda)$ favors planar facets, but strict planarity is not necessarily enforced.

To define the new component of the objective function, DP, we first define a facet f_i as a sequence of connected lines $l_{i,j}$ in the line drawing (as mentioned previously, determining which sequences should be considered planar facets is still not a completely solved problem, see Appendix A for the set of rules we used for the examples in this paper). Each facet adds a term to DP that is a measure of the non-planarity of that facet; it is the following local non-planarity measure summed over the lines in the facet:

$$\left[1 - \left|\frac{(l_{i,j-1} \times l_{i,j}) \cdot (l_{i,j} \times l_{i,j+1})}{\| l_{i,j-1} \times l_{i,j} \| \| l_{i,j} \times l_{i,j+1} \|}\right|\right]^2.$$

Ideally, we would like to find the orthographic extension of the line drawing with the lowest SDA that has exactly planar facets. To achieve this, we use a so-called continuation method (Leclerc 1989), which is a sequence of gradient-descent steps applied to $E(\lambda)$. The sequence begins with the initial condition that Marill suggests ($z = 0$ for all points) and with $\lambda = 0.5$. Then, λ is decreased by a given amount and the gradient-descent algorithm is applied anew, starting at the solution found for the previous value of λ. This is repeated until λ is sufficiently close to zero so that no additional changes occur with further reductions in λ.

Why not simply start with λ close to zero in the first place? The reason is that when λ is sufficiently close to zero, the local minima of $E(\lambda)$ are determined only by the planarity component. Thus, simply starting with λ close to zero would not allow us to find solutions with low SDAs (in fact, when $\lambda = 0$, the original line drawing, which is planar, is a local minimum of $E(\lambda)$). Although we cannot affect the shape of $E(\lambda)$ when λ is small, we can choose the starting point for the gradient-descent algorithm. Thus, the purpose of the continuation method is to choose a sequence of starting points that are first strongly influenced by the SDA term, but which eventually become dominated by the DP term. The method is not guaranteed to find a global minimum of the objective function, but has yielded excellent answers for all the examples discussed in this paper.

3.1 Results

Figures 2—6 illustrate the results of our augmented MSDA algorithm, and allows one to compare them with both Marill's reconstructions and the original 3-D objects that were used to generate the line drawings. The reconstructions are illustrated both graphically (as two views in the upper third of each figure) and in tabular form in the lower third. The first column of the table lists the z coordinates of each object, the second column is the range of lengths of the lines of each object, the third column is the mean and range of the angles formed by all line pairs meeting at a common vertex, the fourth column is the standard deviation of angles (SDA) of each object, and the fifth column is the deviation from planarity (DP) of each object. To simplify the comparison of the results, the recovered z coordinates have been normalized so that the first point always has $z = 0$, and the second coordinate is always positive (this normalization procedure has no effect on the objective function).

We also applied our algorithm to Examples A through I from Marill's paper. Since his algorithm produced approximately planar-faceted solutions by itself in all cases but Example I, it isn't surprising that our algorithm produced solutions almost identical to his. The greatest deviation from his result was for Example I, because Marill's algorithm recovered a significantly non-planar face for the leftmost face of the line drawing.

In all of the examples, the Δzs used by Marill's algorithm (both as a stand-alone algorithm and within the continuation-method) were 0.125, 0.0625, 0.03125, 0.015, and 0.007. We used a smaller initial Δz than Marill suggests because the larger one often forced the algorithm out of the valley of attraction of the current local minimum. Decreasing Δz by a factor of two generally generally allowed the algorithm to run in the fewest number of iterations. Using a smaller final Δz allowed the algorithm to produce significantly more accurate solutions. In the continuation method, λ was started at 0.5, and was decreased by a factor of two a total of four times.

Example J (Figure 2) illustrates Marill's reconstruction for a line drawing of a regular hexagonal prism. This reconstruction not only appears psychologically implausible from these two views, but, as we discuss in the following section, the reconstructed object does not appear rigid when rotated in real-time. It would appear that at least part of the reason for this result is that the recovered facets are clearly non-planar, as shown by the value of DP in the table. Our reconstruction is almost identical to the original hexagonal prism.

In Example K we see that the MSDA principle itself is ambiguous for simple line drawings. Marill's reconstruction takes the line-drawing of a planar hexagonal plate (SDA=0.0) and reconstructs a non-planar object, also with SDA=0.0. By enforcing planarity, however, our reconstruction is quite close to the original hexagonal plate.

In Examples L and N we see further evidence that the MSDA principle by itself is inadequate for even simple line drawings. In both examples, Marill's reconstruction has a significantly lower SDA than the original object, and we consider both of these reconstructions to be psychologically implausible. Our reconstruction of Example L is quite close to the original object, modulo an additive constant and flip of the z-coordinates of the second object (which is invisible to the objective function). Example N is a fairly ambiguous figure, and our reconstruction favored a "hinge" with all angles close to ninety degrees (the original object had a "hinge-angle" of forty-five degrees). Because of the ambiguity of the figure, there exists a family of reconstructions that we consider psychologically plausible, including ours.

Example M shows the reconstruction of a figure for which some of the planar facets are not equiangular. Again, because some of the facets had more than four sides, Marill's algorithm failed to recover a psychologically plausible object. Our reconstruction is reasonably good, but it did adjust the right-angles in the large face by as much as thirteen degrees in order to make the angles in that face closer to being equal. Nonetheless, we consider the reconstruction to be psychologically plausible.

4 Implications for Human Vision

Line drawings provide an effective means of communication about the geometry of 3-D objects. It is a matter of some debate as to whether the interpretation of line drawings is a learned skill, or whether line drawings are isomorphic to some intermediate construction of the human visual system (HVS) in its normal processing of imagery, but in either case an understanding of how humans interpret line drawings is extremely important in enabling man-machine communication with respect to images, diagrams, and spatial constructs. In this section we address two related questions arising out of the investigation described earlier in this paper: (a) under what conditions is a line drawing actually given some intended 3-D interpretation, and (b) under what conditions does a moving rigid (wire frame) object actually appear rigid.

Some, but not all, line drawings are perceived by human subjects as being three dimensional. What attributes of the drawing promote such an interpretation, and what are the constraints on the nature of the resulting 3-D construction? Partially because human introspection is involved, this is a very difficult question to answer. For example, if the drawing is recognized as a known or previously encountered 3-D object, it might be visualized this way even though it violates conditions necessary for an unfamiliar object to be perceived as being three dimensional. Gestalt psychologists have suggested that if the drawing offers a simpler construct when seen as three dimensional, than when seen as being flat, it will be perceived as being three dimensional; however, an effective computational procedure to evaluate "simpler" has yet to be provided (and there is also the problem of producing the corresponding 3-D construct).

It appears to be much more productive to show a human subject a candidate 3-D reconstruction and ask if it corresponds to some given line drawing than it is to tabulate introspective judgments about whether objects appear to be 2-D or 3-D. The former approach, in fact, is how Marill presents his results to the reader. Obviously, he can't show an actual 3-D reconstruction, but only a projection. If he showed the reconstructed object projected without some spatial relocation, then all we have is the original line drawing back again – and no determination can be made; Marill shows two projections of his reconstructed objects, rotated by a few degrees, for evaluation by the reader. Now we know that every orthographic extension is a *geometrically* feasible reconstruction, so on what basis does the human judge acceptability (i.e., what we have called a *psychologically* feasible reconstruction). It is easy to hypothesize a whole list of conditions that should be met – mostly different instantiations of the idea that regularities (such as parallel lines or equal angles and lengths) observed in the line drawing are not accidental, and should be preserved in the reconstructed object; orthographic projective invariants, such as parallelism, should then also be preserved in the reprojections of the spatially relocated object. One could write computational procedures to search for such invariants, but this approach seems incompatible with the universality of the human evaluation process (e.g, none of the invariants we happened to think of may be present in the instances we are considering). A more powerful idea is to require that the computational procedure that produced the original reconstruction give the same result when applied to any of its reprojections – i.e., a consis-

tency criterion.[7] This is exactly the condition that obtains when we observe a moving or rotating object to be rigid; when we see a (continuous) sequence of projections that we perceive as being isomorphic to the same geometric reconstruction, we perceive the object as being rigid.

If we apply the above ideas to an evaluation of Marill's results, we find his algorithm often fails to recover even a geometrically similar object from two different orthographic projections (see Figure 7). Further, when we use the computer to create a rotating display of some of the reconstructions obtained with the use of Marill's algorithm, we see what appears to be the movement of a non-rigid object (see Figure 8). Those reconstructions that have pronounced non-planar facets appear to be non-rigid, while objects with only planar facets appear rigid under rotation. Thus augmenting the MSDA objective function with a planarity term is a necessary step in improving performance in the recovery task. We showed in an earlier section that a "continuation method" based optimization procedure can be easily formulated to bias the search for a MSDA solution which also emphasizes the formation and retention of planar facets; this algorithm gives psychologically plausible answers for all of the examples we have tested.

5 Discussion

Traditional blocks world problems are mathematical in nature, they deal with issues of existence and consistency based strictly on geometric considerations; they make no reference to what people actually see. The problem defined by Marill is psychological; since every line drawing has an infinite number of mathematically valid orthographic extensions and no invalid ones, on what basis does the HVS select a particular extension as being psychologically acceptable? Marill proposed an intriguingly simple criterion for duplicating human preference, but we have shown that, while it often produces an acceptable answer, it is unreliable even in very simple situations.

Marill's work has similarities to the Huffman-Clowes-Waltz approach which focused on how polyhedral vertices can appear in a line drawing, and hence, the constraints such vertices impose on the implied 3-D model; Marill only considers the constraints implied by line-intersections at specified vertices in the line drawing. Mackworth, Kanade, and Sugihara found it necessary to introduce constraints based on the explicit assignment of vertices to planar faces. In this paper we show the need for introducing a similar explicit requirement for planarity (actually, in the context of optimizing an objective function, our constraint is soft in that it can be violated). However, in our case, the requirement for planarity is justified on psychological grounds rather than for achieving a geometrically more competent algorithm.

The preference of the HVS to interpret a line drawing as the most symmetric polyhedral (planar faced) object consistent with the drawing is well established in the psychological literature. Marill appeared to have discovered a simple computational procedure for finding such solutions for any given line drawing,[8] but on closer examination, it became apparent that his MSDA principle does not enforce (or even prefer) planar solutions. Because of this deficiency, MSDA is unreliable even in very simple situations. We were able to prove (Appendix D) that if a planarity preference is explicitly added to the MSDA objective function, then indeed, the non-obvious preference for symmetric solutions is also present. However, we are now forced to address the problem of how to provide the auxiliary information necessary to partition the drawing into the coherent components corresponding to the 3-D planar facets. It appears that the HVS selects some subset of the contours in the line drawing as corresponding to the planar facets in the 3-D model, and if we do not supply this information to a recovery algorithm (either explicitly or by providing a set of conditions implying the same information), we will fail to recover psychologically acceptable models.

Most of the work in the blocks world tradition employed perfect *labeled* line drawings with the assignment of vertices to faces given as part of the input specification. If we follow the same approach (although we are not concerned with having perfect line drawings since our recovery method employs optimization which can tolerate deviations from any of the constraints embodied in the objective function), then we at least have provided a tool for simplifying man-machine communication using the language of line drawings. However, there is obvious theoretical value in understanding the criterion for human selection of the circuits in the line drawing that correspond to planar facets in the 3-D model.[9] In part, this importance is related to the issue of how the HVS recovers the shape of a moving object. Even though there are a few well known exceptions, it is widely believed that the HVS will assume an object to be rigid and correctly recover its shape if this is indeed the case.[10]

[7] The successive reconstructions are not independent; to the extent that they allow a range of interpretations, the parameters selected for one interpretation will influence the parameter selections for successive interpretations.

[8] Marill, of course, only returns the wire frame. But in the case of a Blocks World object, competent algorithms exist for finding all the valid completions of the wire frame as a solid polyhedral object (Strat 1984, Markowsky and Wesley 1981).

[9] As noted in Appendix A, we have made some initial progress toward the solution of this problem and have developed an algorithmic procedure that can successfully handle all of the examples discussed in this paper; but we recognize that this is still far short of a complete solution.

[10] For example, by using Ullman's result that three distinct orthographic projections of four non-coplanar points in a rigid configuration are sufficient to uniquely determine the structure and motion up to a reflection about the image plane.

However, the rigid wire frames with non-planar facets provide a whole class of counter-examples to this belief – they appear to be non-rigid when observed in motion (even at very low speeds where maintaining correspondence of vertices from one projection to the next is no problem). The non-rigidity appears to result from the HVS making incorrect decisions about how the drawing can be partitioned into planar facets (see Appendix E).

6 Summary

Marill's recently published paper claimed that the simple procedure he described could duplicate human judgement in recovering the 3-D wire frame geometry of objects depicted in line drawings. He provided some impressive examples, but no theoretical justification to back his claims. In this paper, we critically examined the merits of Marill's algorithm, provided at least a partial explanation for its competence, identified weaknesses, showed how it could be improved, and discussed the implications of this work for clarifying some important problems in human perception.

In particular, we provided a number of theorems which show that minimizing the standard deviation of angles is (potentially) a simple and effective method for selecting symmetric solutions when the constraining line drawing (which is the projection of a wire frame that may be incomplete) permits such interpretation. On the other hand, we showed that Marill's algorithm could fail in simple cases, that he employed an optimization procedure that was often too weak to find the correct answer even when it was within the competence of the objective function, and that the algorithm would often produce wire frames with non-planar facets (something no human would intuitively accept in perceiving a straight-line drawing as a 3-D configuration).

We argued that an important condition in testing or evaluating the psychological correctness of a reconstruction is that its reprojections (after spatial relocation) result in the same object being produced by the recovery algorithm. For the human visual system, this is equivalent to the condition that the recovered object appear rigid when observed during movement or rotation. The perception of rigidity for wire frames appears to be highly correlated with the presence or absence of strongly non-planar facets. By modifying Marill's objective function to explicitly favor planar-faceted solutions, and by using a more competent optimization technique, we were able to demonstrate significantly improved performance in all of the examples Marill provided as well as those additional ones we constructed ourselves.

References

Clowes, M. B. (1971). On seeing things. *Artificial Intelligence, 2(1)*, 79–116.

Draper, S. W. (1981). The use of gradient and dual space in line-drawing interpretation. *Artificial Intelligence, 17*, 461–508.

Huffman, D. A. (1971). Impossible objects as nonsense sentences. In Meltzer and Michie, editors, *Machine Intelligence 6*, pages 295–323, Edinburgh Univ. Press.

Kanade, T. (1980). A theory of origami world. *Artificial Intelligence, 13(1)*, 279–311.

Leclerc, Y. G. (1989). Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision, 3(1)*, 73–102.

Mackworth, A. K. (1973). Interpreting pictures of polyhedral scenes. *Artificial Intelligence, 4(2)*, 121–137.

Marill, T. (1991). Emulating the human interpretation of line-drawings as three-dimensional objects. *IJCV, 6(2)*, 147–161.

Markowsky, M. A. and Wesley, G. (1981). Fleshing out projections. *IBM J. R&D, 25(6)*, 934–954.

Strat, T. M. (1984). Spatial reasoning from line drawings of polyhedra. In *DARPA Image Understanding Workshop*, pages 230–235.

Sugihara, K. (1982). Mathematical structures of line drainings of polyhedrons—toward man-machine communication by means of line drawings. *IEEE PAMI, 4(5)*, 458–469.

Sugihara, K. (1984). A necessary and sufficient condition for a picture to represent a polyhedral scene. *IEEE PAMI, 6(5)*, 578–586.

Waltz, D. A. (1972). *Generating semantic descriptions from line drawings of scenes with shadows.* Technical Report AI-TR-271, MIT.

Whitely, W. (1986). Two algorithms for polyhedral pictures. In *Proc. Second Annual Symp. on Computational Geometry*, pages 142–149.

Appendices

A Psychological Assumptions

The following are some of the basic assumptions that we believe are typically made by people in the reconstructions of wire frames from line drawings, and some constraints relevant to partitioning a line drawing into planar facets. They are are known to have rare exceptions.

1) Three dimensional wire frames, derived from line drawings, have implied planar faces inside subsets of their closed circuits; they *can* also have struts, such as legs or bracing wires, in or on a planar face. (Strongly non-planar facets produce psychologically unacceptable solutions.)

2) Symmetric reconstructions are preferred over non-symmetric ones.

3) Parallel lines in a line drawing are parallel in space. Lines connecting vertices falling on two parallel lines are in a common plane with the two parallel lines.

4) Many-sided convex closed contours without internal *circuits* (in a 2-D line drawing) are likely to correspond to the contours of planar faces in the corresponding 3-D orthographic extension (see B4).

5) A closed simple contour in a line drawing, without internal lines, corresponds to a planar face in the corresponding 3-D reconstruction.

An algorithmic procedure for identifying 3-D planar facets in the corresponding 2-D line drawing was constructed by composing the requirements of items 3, 4, and 5 into a single algorithm. This procedure is sufficient to deal with all of the examples we discuss in this paper, but is not general enough to handle other cases we can think of.

B Projective Invariants

The following are some important projective invariants for planar geometric structures.

1) The sum of the interior angles (measured between 0 and 360 degrees) of a closed planar contour with n sides equals (n-2)180 degrees. Thus, since a polygon of n sides projects to a polygon of n sides under both orthographic and central projection, the mean value of the interior angles of a given closed planar contour [(n-2)180/n] is *invariant under both orthographic and central projection.*

We note that Marill measures angles only in the interval between 0 and 180 degrees. To the extent that we are primarily concerned with equiangular closed contours in the application of the above theorem in explaining and using his results, this discrepancy is irrelevant since all the interior angles of such contours are less than 180 degrees.

2) Consider an angle (two line segments sharing a common endpoint) in 3-D space and its orthographic projection. We will call the plane containing the angle the source plane, and the plane containing its projection the projection plane. If the angle is translated in the source plane, its projection is also translated, but does not change in magnitude from its original projected value. Now consider a set of n angles lying on a common source plane, such that the sum of these angles is 360 degrees. If it is also the case that the angles can be translated so that when all their vertices coincide, they exactly span an angle of 360 degrees, then the mean value of the set of angles (360/n) is unaltered under orthographic projections We will call such a collection of angles a "complete-star."[11]. We note that if an essentially infinite number of copies of an angle of d degrees (where 360/d = k and k is an integer) is uniformly distributed in orientation over a plane, then the mean value of the angles under any orthographic projection of the plane is the constant value d.

3) We note that if the angle between two line segments is less than 180 degrees, the angle can be closed to form a triangle, and since triangles are preserved under both orthographic and central projection, an angle of less than 180 degrees will never transform under such projections into one of more than 180 degrees. We will call a closed planar contour convex if the region it bounds is convex. Since a convex contour has all internal angles of less than 180 degrees, a convex planar contour remains convex under both orthographic and central projection.

4) We note that the orthographic projection of an arbitrary non-planar polygonal space curve, with four or more sides, has a probability of projecting to either a non-simple or concave curve with a probability (P) that increases with the number of sides:

$$P > 1 - .5^{n-3} \quad for\ n \geq 4$$

This expression is based on the following model: Consider a process that generates a chain of 3-D random vectors by generating three random numbers for each vector (in spherical coordinates, an angle uniformly distributed between 0 and 360 degrees, a second angle between 0 and 180 degrees, and a length uniformly distributed between 0 and some fixed integer L). As each vector is generated we extend the projection of the developing space curve on the X-Y image plane. The process stops after some fixed number of steps which is determined by choosing a random number in some given range; the curve is now closed by connecting the starting point, which could be the origin of the X-Y plane, to the last point generated and this determines whether the inside is to the left or right as we follow the chain of edges of the projected polygon. We note that the only relevant factor in

[11] Example C, for instance, contains a complete-star consisting of the eight 45 degree angles formed at the corner vertices by the diagonals with the sides of the square. Example G contains this same configuration in its central plane.

whether the projected closed contour is convex or concave is the cylindrical angle giving the rotation of each of the random vectors relative to the X axis in the image plane. For more than three sides, there is a 50% probability at each vertex that the inside angle is greater than 180 degrees which thus produces a concave polygon (the last closing side can be ignored since it does not have the same statistics as the other edges in our random model). Other probabilistic models would give non-identical but similar results. The > condition is based on additional considerations, such as the projected curve intersecting itself even though the input specification does not record a vertex at the cross point.

5) Closed 4-sided polygonal space curves with 90 degree angles at each vertex are planar contours. To prove this assertion, let the sequence of vertices be labeled a, b, c, and d. Let the plane containing lines L_{ab} and L_{bc} and thus vertices a, b, and c) be called P_1. Since all angles are 90 degrees, L_{cd} must lie in a plane (P_2) normal to L_{bc} at c. Similarly, L_{ad} must lie in a plane (P_3) normal to L_{ab} at a. Vertex d must then lie on the line (L_d) of intersection of P_2 and P_3 which is normal to P_1. We know one solution is to locate d at the point of intersection (d^*) of L_d and P_1 (where a, b, c, and d^* form a rectangle). This is the planar solution and we wish to show that no other solution is possible. We note that a second constraint on the location of d is that it must lie on a sphere with diameter ac (i.e., all right angles, with legs passing through points a and c, must be inscribed angles of circles through a and c with diameter ac). We know d^* lies on the sphere and P_1 is a bisecting plane of the sphere. Thus L_d is tangent to the sphere at d^* and d^* is the only possible solution.

C A Partition Theorem

The variance of a set S of n objects $\{a_i\}$ is defined as:

$$V = \frac{1}{n}\sum_{i=1}^{n}(a_i - M)^2 = \left[\sum_{i=1}^{n} a_i^2/n\right] - M^2$$

where:

$$M = \frac{1}{n}\sum_{i=1}^{n} a_i$$

Let us now partition the $\{a_i\}$ into k subsets, such that subset $\mathbf{S_j}$ has n_j elements and mean M_J where:

$$M_j = \frac{1}{n_j}\sum_{\mathbf{S_j}} a_i$$

Let V_j be the variance of $\mathbf{S_j}$ about M_J and let $\Delta_j = (M - M_j)$.

Theorem:

$$V = \frac{1}{n}\sum_{j=1}^{k} n_j[V_j + \Delta_j{}^2]$$

Proof: The expression for V can be rewritten as:

$$V = \frac{1}{n}\left[\sum_{\mathbf{S_1}}[a_i - (M_1 + \Delta_1)]^2 + \sum_{\mathbf{S_2}}[a_i - (M_2 + \Delta_2)]^2 + \cdots + \sum_{\mathbf{S_k}}[a_i - (M_k + \Delta_k)]^2\right]$$

If we let:

$$V_j^* = \sum_{\mathbf{S_J}}[a_i - (M_j + \Delta_j)]^2$$

Then we have:

$$V_j^*/n_j = \sum a_i^2/n_j + M_j^2 + \Delta_j^2 - 2\Delta_j\sum a_i/n_j - 2M_j\sum a_i/n_j + 2M_j\Delta_j$$

Given that $\sum a_i/n_j = M_j$, we note that the 4th and 6th terms cancel and the 2nd and 5th terms combine:

$$V_j^*/n_j = \left[\sum a_i^2/n_j - M_j^2\right] + \Delta_j^2 = V_j + \Delta_j^2$$

And:

$$V_j^* = n_j[V_j + \Delta_j^2]$$

QED

D A Symmetric Preference Theorem

Recall that.

1) In Appendix B we showed that the average angle of all planar orthographic extensions of a given simple closed 2-D contour are the same, and that the average angle of all planar orthographic extensions of a complete-star are also the same.

2) In Appendix C we proved a theorem that allows us to compute the SDA of a set of simple closed planar contours (and/or complete-stars) as the sum of two components. The first component is the variance of the angles in a contour or star about the mean angle of that contour or star, summed over all contours and stars. The second component is a weighted sum of the squared differences between the mean angle of each contour and star, and the average of all the angles under consideration.

By 1), the second component of the variance is constant over all planar orthographic extensions because (a) the mean of each contour and star is constant over all such extensions, and (b) the mean of all angles can be computed as the weighted sum of the mean of each contour and star.

Consequently, if we restrict our attention to the planar orthographic extensions of a line drawing, then by 2) above only the first component of the variance will change over the extensions. Since the first component is zero for an extension comprising only equiangular planar contours and stars (such as the solutions for Examples A, B, C, G, J, K, and L), and since it is positive otherwise, then such symmetric solutions correspond to the global minimum of the SDA over all planar orthographic extensions.

E Factors Affecting The Perception Of Non-Rigidity

If we rotate a *randomly* derived orthographic extension of almost any of the line drawings used as examples in this paper, the object appears non-rigid to most observers (even though, of course, the wire frame is actually a rigid object). While there are many possible explanations for this phenomenon, our conjecture is that it is primarily due to special position projections of the wire frame (that occur at one or more poses in its rotation) that lead the HVS to incorrectly assume that some projective invariant (such as parallel lines, see Figure 8) is being observed. This, in turn, causes incorrect expectations about the presence and location of planar facets.

We informally looked at some other possible causative factors, but did not observe consistent non-rigidity phenomena. For example, we looked at objects, such as Example N that produce compelling 3-D interpretations with Necker reversals, but for which the drawing is incomplete—it does not show all the edges that should be visible, e.g., where planar faces intersect. There was the possibility that these missing edges in the 3-D model (and thus missing lines in the drawing) could cause the appearance of a non-planar faceted object to be observed. But the hinge, and the few other objects we looked at in this category, appeared rigid.

We also looked at non-planar orthographic extensions of drawings that generally appeared flat, including blocks world type drawings that do not have corresponding polyhedral realizations (such as example O). The results here were ambiguous. The rotating objects generally produced illusions of non-rigidity, but since these objects did not always appear three-dimensional, the illusions were generally very weak.

Some other casual experiments include cases where all the lines connecting the vertices of the wire frames are deleted, we then observed that some of the wire frames that originally appeared non-rigid, now appeared to be rigid under rotation. And, as a general observation, we have not encountered any examples in which the wire frame of a blocks world object appears non-rigid when in motion.

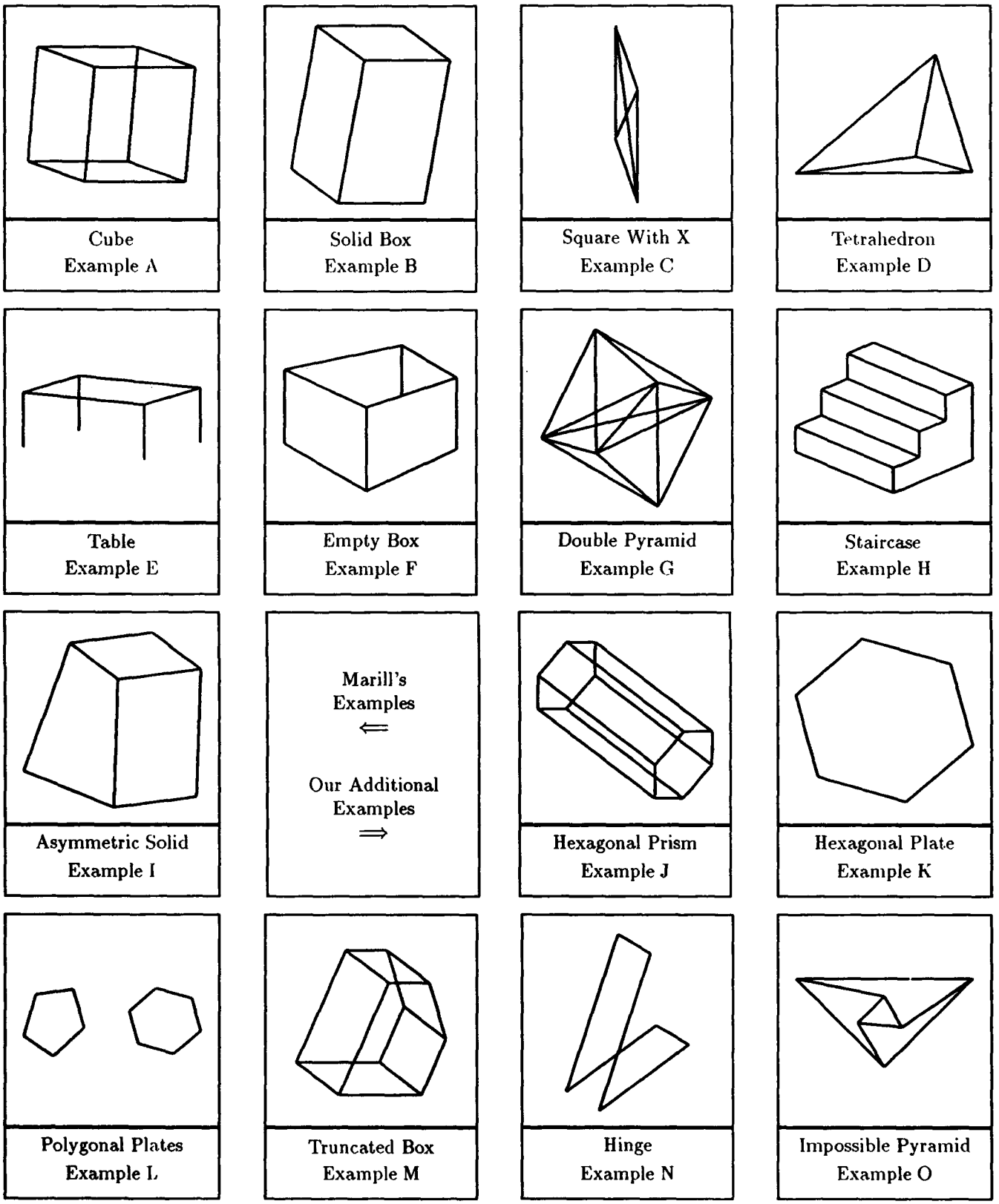
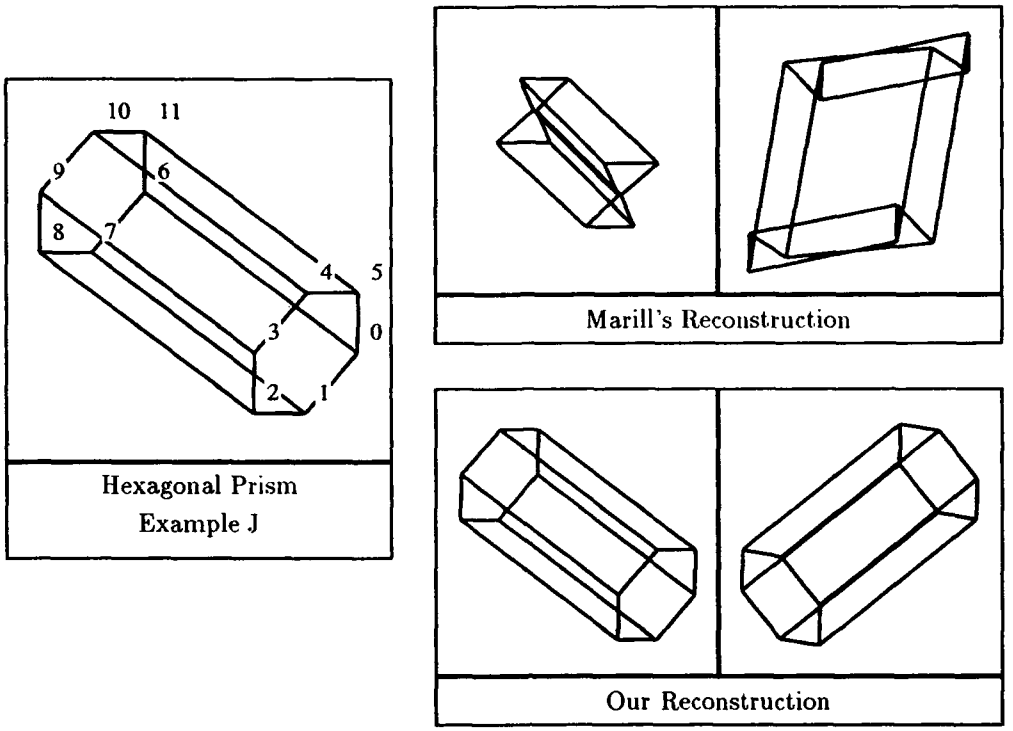


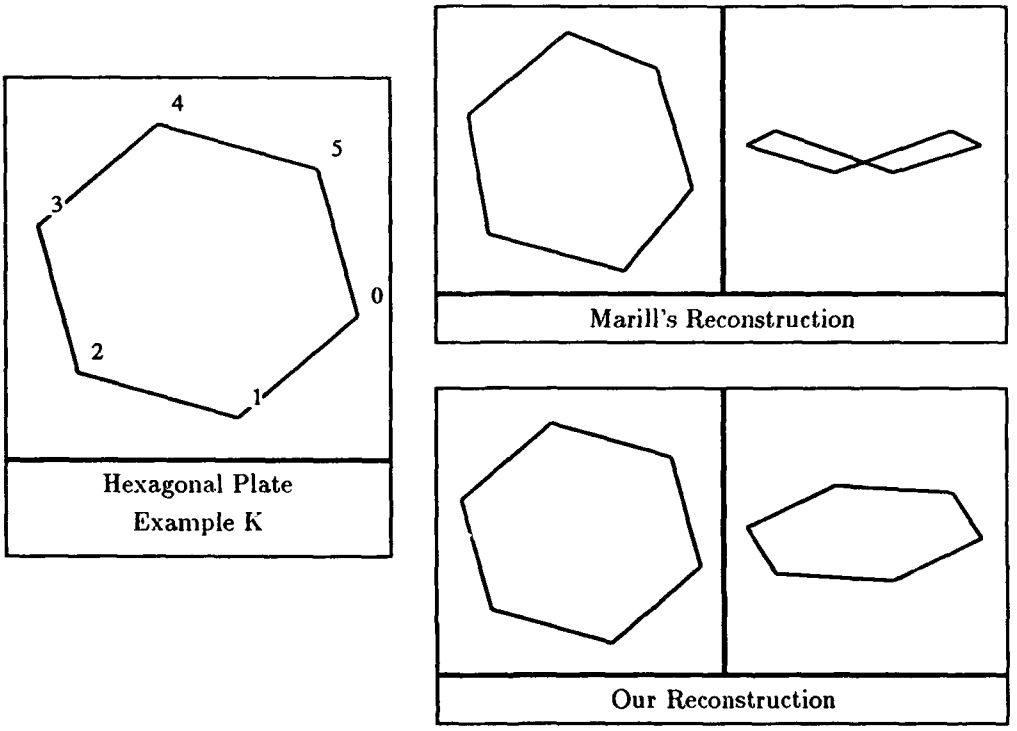
Figure 1: The line drawings examined in this paper. Examples A through I are taken from Marill's paper. Examples J through N are line drawings introduced in this paper for which Marill's algorithm failed to recover a psychologically plausible 3-D model. Example O is a line drawing for which a psychologically plausible 3-D model is not feasible.



Points	(1.97 -1.00) (1.32 -1.75) (0.67 -1.75) (0.68 -1.00) (1.34 -0.25) (1.98 -0.25) (-0.68 1.00) (-1.34 0.25) (-1.98 0.25) (-1.97 1.00) (-1.32 1.75) (-0.67 1.75)
Lines	(0 1) (1 2) (2 3) (3 4) (4 5) (5 0) (6 7) (7 8) (8 9) (9 10) (10 11) (11 6) (0 6) (1 7) (2 8) (3 9) (4 10) (5 11)
Faces	(0 1 2 3 4 5) (6 7 8 9 10 11) (0 1 7 6) (1 2 8 7) (2 3 9 8) (3 4 10 9) (4 5 11 10) (5 0 6 11)

	Zs	Lengths	Angles (Mean / Range)	SDA	DP
Original Object	0.00 0.10 0.87 1.53 1.43 0.66 -2.23 -2.12 -1.36 -0.69 -0.80 -1.57	1.0 to 4.0	100.0 90.0 to 120.0	0.060923	0.000000
Marril's Reconstruction	0.00 0.42 -3.02 -2.37 -3.04 0.68 -0.51 0.16 -3.56 -2.87 -3.29 0.14	1.0 to 3.8	84.2 48.4 to 109.9	0.109002	0.035421
Our Reconstruction	0.00 0.11 0.92 1.58 1.47 0.68 -2.09 -1.98 -1.18 -0.51 -0.62 -1.43	1.0 to 3.9	100.0 89.0 to 121.0	0.061034	0.000000

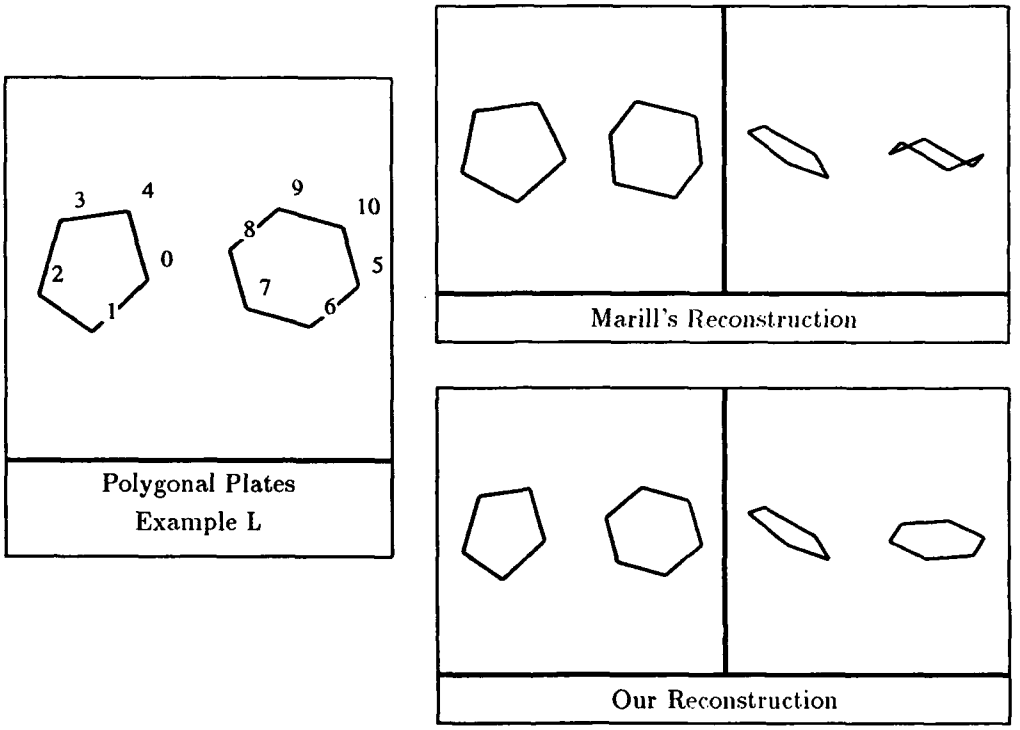
Figure 2: Example J. This line drawing was created by orthographically projecting a specific 3-D wire frame object. In this case, the object was a regular hexagonal prism. Although arbitrary line drawings can be used as input to the reconstruction algorithms described in this paper (with greater or lesser success in reconstruction), all of the examples introduced in this paper were created by starting with specific 3-D objects. The panels in the upper right show two views of the object reconstructed by Marill's algorithm. The first view is of the object rotated about the vertical axis by 30 degrees, and the second is of the object rotated about the horizontal axis by 90 degrees. The two panels in the lower right show two views of the object reconstructed by our algorithm. The table below this is the internal representation of the line drawing used by the reconstruction algorithms. Note that intersections such as those between lines (1 7) and (2 3) are not represented. Marill's algorithm uses only the first two components of this representation. The third component is derived from the line drawing using heuristics described in Appendix A. The table at the bottom shows the results of the reconstructions in written form.



Points	(0.96 -0.27) (0.24 -0.89) (-0.72 -0.61) (-0.96 0.27) (-0.24 0.89) (0.72 0.61)
Lines	(0 1) (1 2) (2 3) (3 4) (4 5) (5 0)
Faces	(0 1 2 3 4 5)

	Zs	Lengths	Angles (Mean / Range)	SDA	DP
Original Object	0.00 0.32 0.24 -0.15 -0.48 -0.40	1.0 to 1.0	120.0 120.0 to 120.0	0.000000	0.000000
Marril's Reconstruction	0.00 0.22 -0.12 -0.00 0.22 -0.12	0.9 to 1.1	116.2 116.2 to 116.2	0.000000	0.030363
Our Reconstruction	0.00 0.34 0.28 -0.09 -0.42 -0.31	1.0 to 1.0	120.0 119.7 to 120.3	0.000013	0.000000

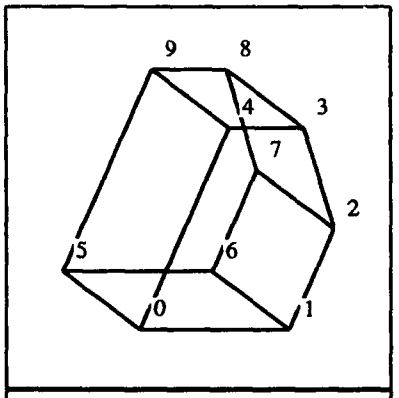
Figure 3: Example K.



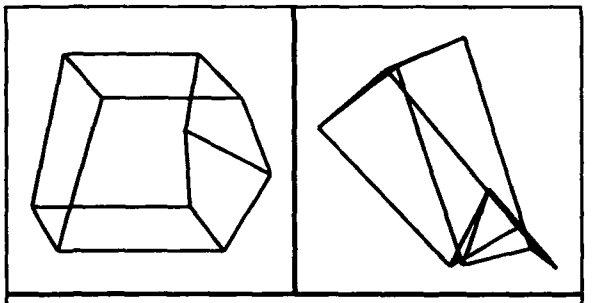
Points	(0.81 -0.19) (-0.02 -0.96) (-0.82 -0.41) (-0.49 0.71) (0.52 0.85) (3.96 -0.27) (3.24 -0.89) (2.28 -0.61) (2.04 0.27) (2.76 0.89) (3.72 0.61)
Lines	(0 1) (1 2) (2 3) (3 4) (4 0) (5 6) (6 7) (7 8) (8 9) (9 10) (10 5)
Faces	(0 1 2 3 4) (5 6 7 8 9 10)

	Zs	Lengths	Angles (Mean / Range)	SDA	DP
Original Object	0.00 0.29 0.94 1.06 0.47 0.47 0.15 0.23 0.63 0.95 0.87	1.0 to 1.2	114.5 108.0 to 120.0	0.010876	0.000000
Marril's Reconstruction	0.00 0.29 0.91 1.03 0.47 0.47 0.16 0.72 0.47 0.78 0.23	0.9 to 1.2	108.0 107.7 to 108.3	0.000007	0.165552
Our Reconstruction	0.00 0.29 0.93 1.05 0.47 0.44 0.79 0.72 0.36 0.02 0.09	1.0 to 1.2	114.5 107.9 to 120.1	0.010843	0.000001

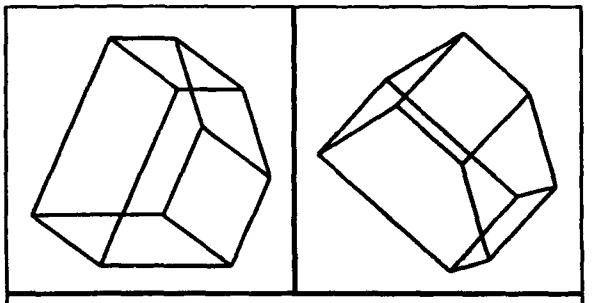
Figure 4: Example L. Note that Marill's unacceptable reconstruction has an SDA that is significantly lower than that of the psychologically correct original object. Thus, the MSDA principle itself has failed in this instance.



Truncated Box
Example M



Marill's Reconstruction

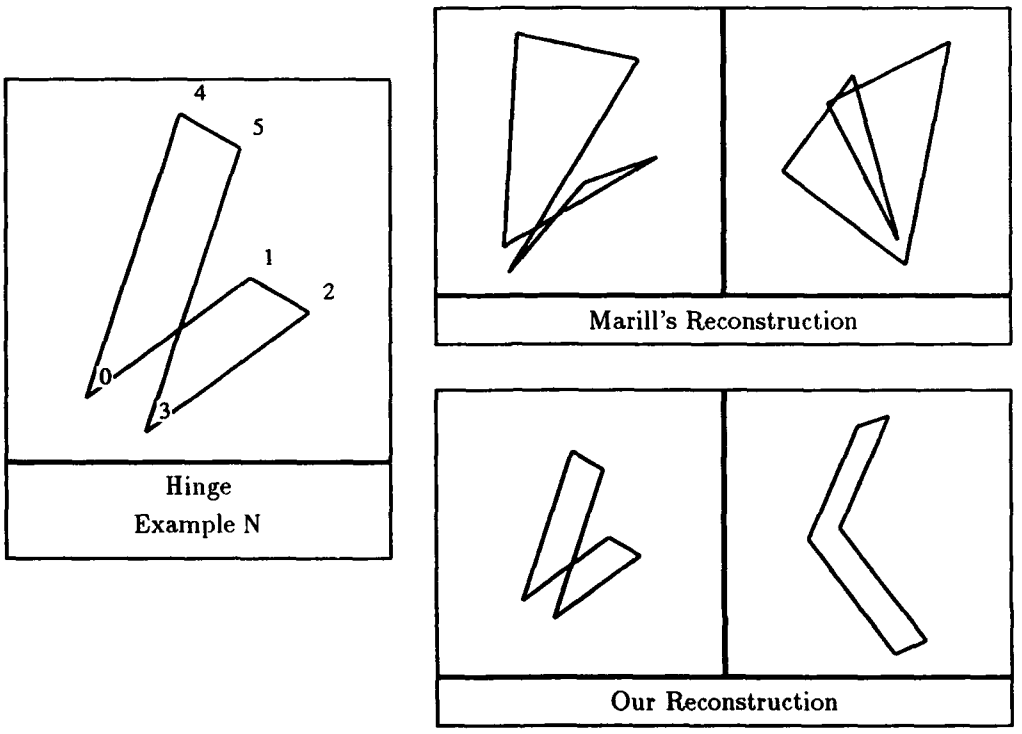


Our Reconstruction

Points	(0.15 -0.06) (0.80 -0.06) (0.99 0.38) (0.86 0.81) (0.54 0.81) (-0.18 0.19) (0.46 0.19) (0.66 0.63) (0.53 1.06) (0.20 1.06)
Lines	(0 1) (1 2) (2 3) (3 4) (4 0) (5 6) (6 7) (7 8) (8 9) (9 5) (0 5) (1 6) (2 7) (3 8) (4 9)
Faces	(0 1 2 3 4) (5 6 7 8 9) (0 5 9 4) (4 3 8 9) (3 2 7 8) (2 1 6 7) (1 0 5 6) (4 0 5 9)

	Zs	Lengths	Angles (Mean / Range)	SDA	DP
Original Object	0.00 0.77 0.61 0.06 -0.32 0.28 1.04 0.88 0.34 -0.04	0.5 to 1.0	96.0 90.0 to 135.0	0.071281	0.000000
Marril's Reconstruction	0.00 0.77 0.97 0.87 -0.16 0.28 0.96 0.58 0.95 -0.01	0.4 to 1.1	86.9 43.0 to 137.1	0.126276	0.094751
Our Reconstruction	0.00 0.57 0.53 0.07 -0.23 0.37 0.94 0.89 0.42 0.13	0.4 to 1.0	95.9 82.0 to 129.1	0.055758	0.000037

Figure 5: Example M. Note that our reconstruction has a slightly lower SDA than that of the original object, indicating the preference of our algorithm for equiangular faces.



Points	(-0.58 0.24) (0.95 1.36) (1.50 1.04) (-0.02 -0.08) (0.30 2.89) (0.86 2.56)
Lines	(0 1) (1 2) (2 3) (3 5) (5 4) (4 0)
Faces	(0 1 2 3) (0 4 5 3)

	Zs	Lengths	Angles (Mean / Range)	SDA	DP
Original Object	0.00 0.64 -0.12 -0.77 -0.47 -1.24	1.0 to 2.8	75.0 45.0 to 90.0	0.137078	0.000000
Marril's Reconstruction	0.00 1.16 -1.61 -0.85 -1.19 0.86	2.0 to 3.3	63.9 63.3 to 64.5	0.000087	0.071300
Our Reconstruction	0.00 2.04 1.80 -0.19 -2.01 -2.19	0.7 to 3.4	89.4 88.0 to 90.1	0.000167	0.000000

Figure 6: Example N. The SDA of Marill's unacceptable reconstruction is again significantly lower than that of the psychologically correct original object.

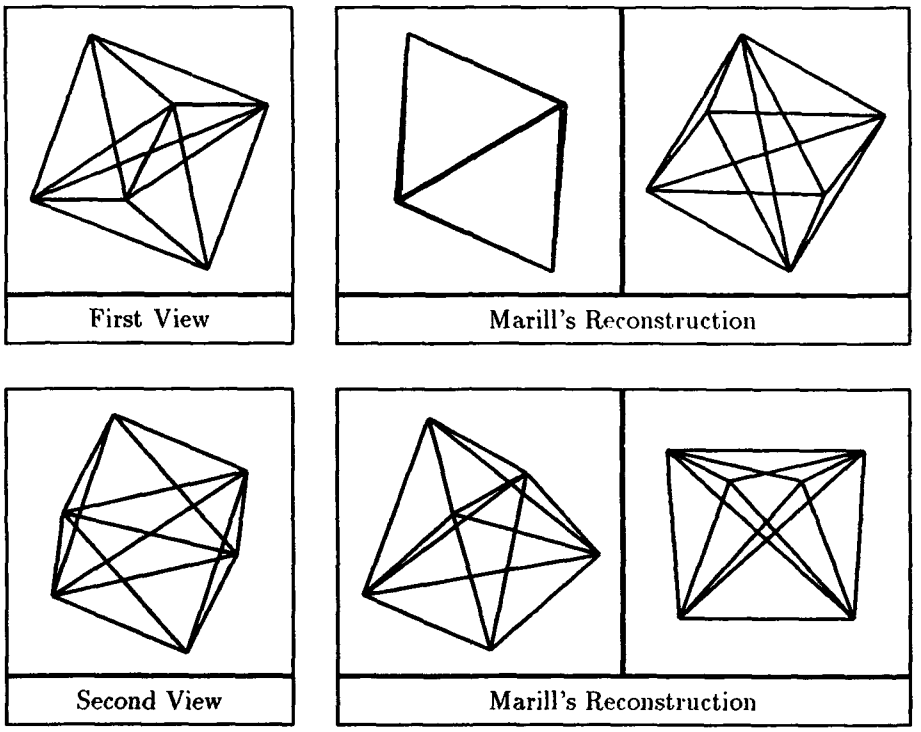


Figure 7: Illustration of the failure of Marill's algorithm to recover geometrically similar 3-D models from two different projections of the same 3-D object. The top row shows the input line drawing of the 3-D object as seen from one view-point (similar to Example G), and two views of Marill's reconstructed object. The bottom row shows the input line drawing of the same 3-D object as seen from a different view-point, and two views of Marill's reconstructed object. The two reconstructed objects not only appear different, but are in fact significantly different geometrically, as we verified by examining their internal representation. In contrast, applying our algorithm to both of these input line drawings, as well as ten other randomly chosen views produced reconstructions with an angular error of less than thirteen degrees from the original object.

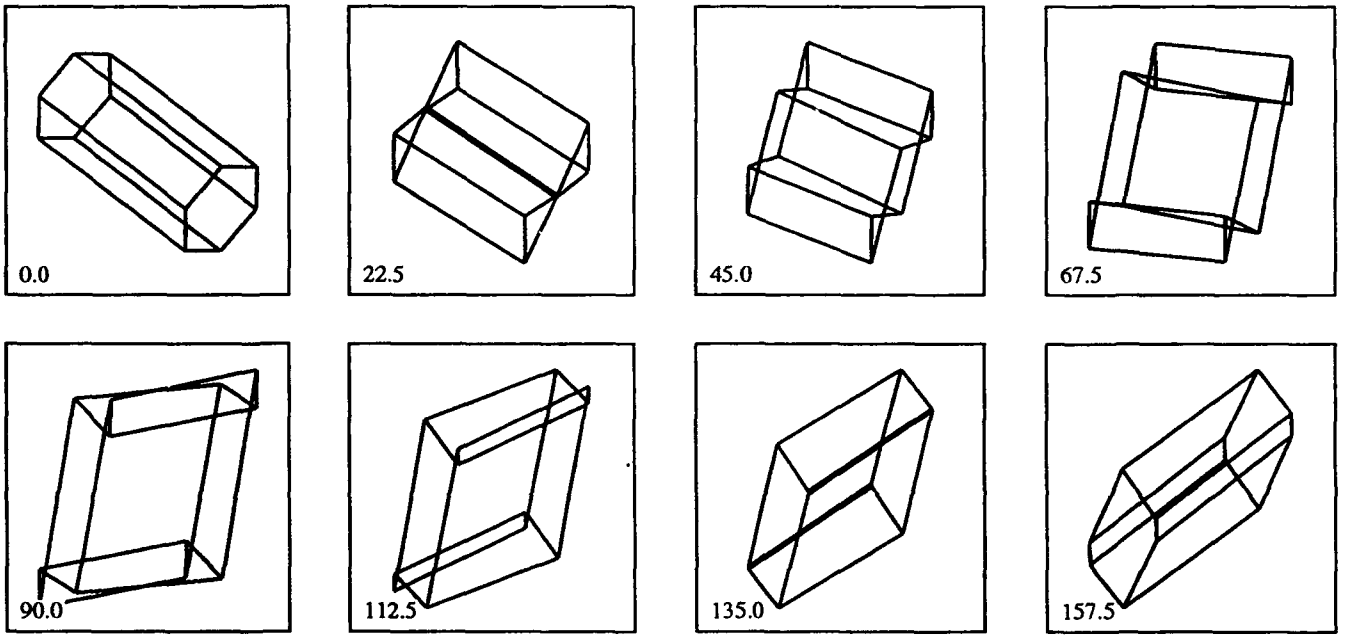


Figure 8: The illusion of non-rigidity for a rotating wire frame with non-planar facets. The wire frame, Marill's reconstruction of Example J, is rotated about a vertical axis in the center of the object. The rotation angle is written in the lower left-hand corner of each box.

Can Too Much Perspective Spoil the View? A Case Study in 2D Affine Versus 3D Perspective Model Matching

J. Ross Beveridge E. M. Riseman

Computer and Information Science Department
University of Massachusetts at Amherst
Amherst, MA. 01003 *

Abstract

At issue is the importance of fully accounting for 3D perspective in 3D model-based robot navigation. A probabilistic combinatorial optimization algorithm searches for an optimal match between landmark and image features by initiating an iterative generate-and-test procedure from a randomly selected set of correspondence mappings. The 2D-to-2D version of this algorithm approximates full 3D perspective with a 2D affine transform - rotation, translation and scale - applied to a 2D projection of the 3D landmark model. A 3D-to-2D version recomputes the robot's 3D pose relative to the model and reprojects the model during matching. In tests, the 3D-to-2D version reliably recovers the robot's true position. The 2D-to-2D version does equally well when initial errors do not introduce perspective distortion, and does so in roughly one fifth the time. However, it fails on some cases where perspective effects are pronounced.

1 Introduction

The problem of matching 3D models to 2D image features arises in many domains. Here we consider problems associated with robot navigation. A robot moving through known surroundings tracks its progress using vision, and moving down a hallway it acquires images such as those shown in Figures 1 and 2. It must test and update its position estimate based upon the appearance of known landmarks. To illustrate, Figures 3A and B show a robot in two possible positions. If the robot believes it is at position A, when it is really at position B, then it should correct this error through landmark recognition.

Recognition involves identifying prominent 3D features in an image. In this case, the landmark is a simple, partial, 3D wire frame model of some distinctive features in a hallway. The model can be seen in Figure 3. The image features are 2D line segments extracted from the images using the Burns algorithm [Bur86]. Perspective effects are pronounced in this domain, as Figure 4 illustrates. Figure 4 shows the landmark as it appears to the

*This work was supported by the Defense Advanced Research Projects Agency (via TACOM) under contract DAAE07-91-C-R035 and by the National Science Foundation under grant CDA-8922572.



Figure 1: Image 1 taken at Position A

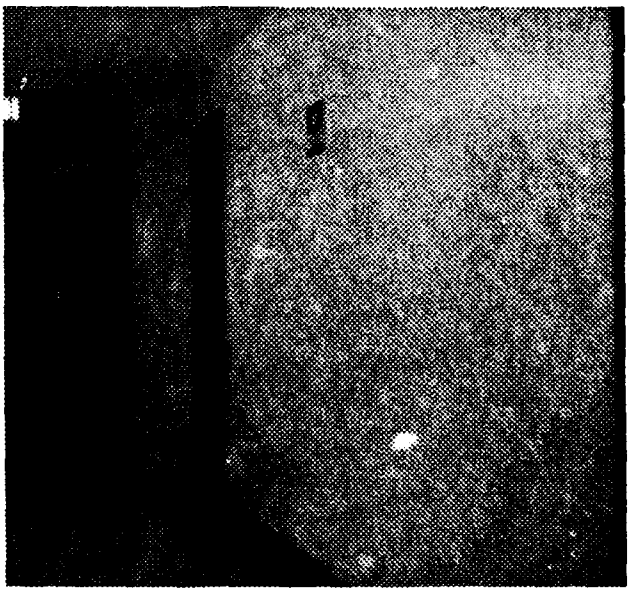


Figure 2: Image 2 taken at Position B

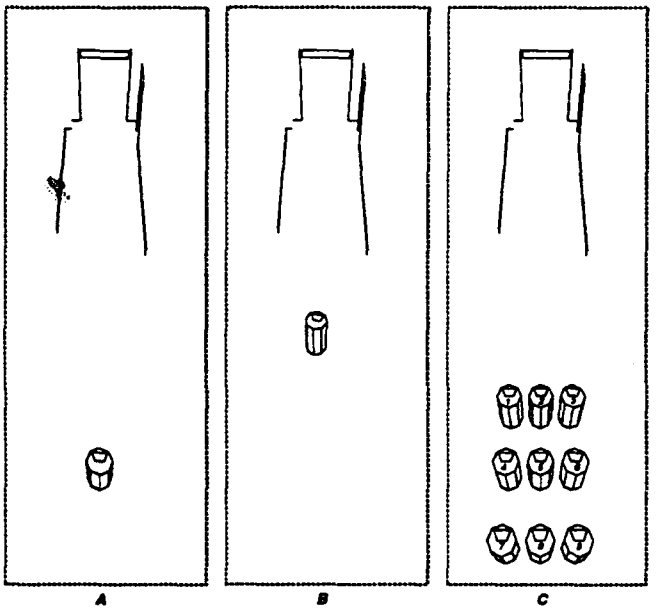


Figure 3: A) Robot in position for Image 1, B) Robot in position for Image 2, C) Nine positions from which to solve for the true position. The 10 3D line segments that make up the landmark are visible in all three examples.

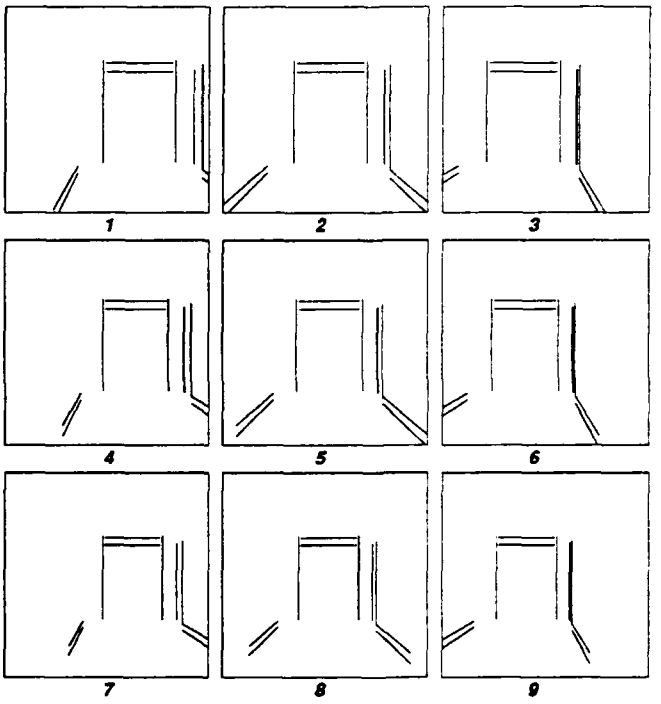


Figure 4: Nine views of landmarks as seen from nine different positions around a nominal location. The relative orientation of the baseboards and the door frame change as the robot's position estimate shifts laterally.

robot from each of the nine poses shown in Figure 3C. Pose refers to the robot's 3D position and orientation relative to the landmark.

At issue is the importance of accounting for 3D perspective while matching landmark features to image features. In side-by-side tests on a set of example problems, the performance of two versions of an otherwise identical matching algorithm will be compared. The resultant match found by each system is used to estimate the robot's true pose using an algorithm developed by R. Kumar [Kum89, Kum90].

The two versions of the matching algorithm are called the 2D-to-2D system and the 3D-to-2D system. In the 2D-to-2D system, a restricted 2D affine transformation - rotation, translation, and scale in the image plane - is used to account for differences between the landmark's estimated and true appearance. This system matches 2D data to a 2D projection of the model generated from the initial pose estimate. We've described this basic approach previously in [Bev90, Fen90].

In the 3D-to-2D system, the landmark model is repeatedly reprojected into the 2D image plane based upon incrementally updated 3D pose estimates. This system uses Kumar's iterative pose algorithm [Kum89, Kum90] during matching.

The discrete search space of possible feature mappings is the same for each side-by-side test. The size of these spaces is staggering. To see this, let M be the set of 3D line segments in the landmark model, let D be the set of 2D data line segments in the image, and let S be the set of model-data pairs which are candidate matches.

$$S \subseteq M \times D \tag{1}$$

In landmark recognition, the initial pose estimate usually constrains the possible pairings between model and data segments, and S is considerably smaller than the complete space of pairs, $M \times D$.

It is not prudent to assume that the correct correspondence between model and data features is one-to-one. The process of extracting image features can lead to fragmentation and accretion. We therefore consider many-to-many mappings, and hence the discrete space of possible correspondences C is the powerset of S.

$$C = 2^S \tag{2}$$

Often S contains 50, 100 or more model-data pairs.

The matching algorithm which searches C is a local search algorithm specifically adapted to model matching [Bev89, Bev90]. In general, a local search algorithm moves from an initial solution, via transformations, to one that is locally optimal [Ker72, Lin73, Pap82]. It is certainly not guaranteed to find the globally optimal match. However, if local search is initiated from independently chosen random starting points, the probability of missing the optimum can be made arbitrarily small by sufficiently increasing the number of trials. This random

sampling strategy is employed here to find an optimal match.

Optimality is defined in terms of a match error. The optimal match, c^*, minimizes this error function:

$$E_{\text{match}}(c^*) \leq E_{\text{match}}(c) \quad \forall c \in C \tag{3}$$

The match error, E_{match}, is a combination of two terms.

$$E_{\text{match}}(c) = E_{\text{fit}}(c) + E_{\text{om}}(c) \tag{4}$$

The first, E_{fit}, is a residual squared error obtained by first fitting the model to the corresponding data. The second, E_{om}, penalizes matches which omit portions of the model from the match. These two forms of error are discussed below.

The experiments presented here test both the importance and associated cost of accounting for perspective during matching. In experiment 1, both the 2D-to-2D and 3D-to-2D approaches are used to estimate the true position of the robot, position A shown in Figure 3A, from each of the nine initial pose estimates shown in Figure 3C. Experiment 2 tests the ability of each system to recover from a confusion between position A and position B.

2 Landmark Matching and Local Search

In our previous work on landmark-based navigation, [Bev90, Fen90], matching was completely separated from 3D pose recovery [Kum89]. The basic scenario was:

1. Project the 3D landmark model features into the image plane using an initial estimate of the robot's pose. Initial estimates are generated by a navigation/planning module [Fen90].
2. Use a rough estimate of the uncertainty in the initial pose to determine a set of candidate model-data pairs, S defined in equation 1.
3. Use the 2D-to-2D matching algorithm to find an optimal correspondence, c^*, between model and data features [Bev90].
4. Use the corresponding landmark and image features in c^* to recover the 3D pose of the robot relative to the landmark. This is done with the iterative algorithm developed by Kumar [Kum89].

The strength of this approach is that it reduces a 3D-to-2D matching problem to a comparatively simpler 2D-to-2D matching problem. The potential weakness is that rotation, translation and scaling in the plane may be insufficient to recover from an initial error in robot position that produces perspective distortion in the projected 2D model. In hallways, where the side walls come toward the camera, perspective effects associated with lateral position error can be dramatic. Figure 4 illustrates the effect. Note the change in relative angle between the baseboards and the doorway.

For 2D-to-2D matching, a model is fit to corresponding data by solving for the rotation, translation and scale which minimizes the integrated, squared, perpendicular distance between 2D data line segments and infinitely extended 2D model lines. E_{fit} is a normalized function of the residual point-to-line squared error after fitting. The best fit 2D pose has a closed form solution involving a simple quadratic equation [Bev90].

The omission error is a function of the percentage of 2D model line segments not covered by data line segments. Coverage is defined in terms of the perpendicular projection of data segments onto model lines. A point on a model line segment is covered when a point on a data segment projects onto it. A more detailed discussion of the omission error appears in [Bev90].

In extending this approach to 3D-to-2D matching, the 2D-to-2D closed form pose computation is replaced by an iterative 3D-to-2D pose computation developed by R. Kumar [Kum89, Kum90]. This algorithm solves for the 3D position and orientation of the camera relative to a model by minimizing a sum of squared 3D point to plane distances. The points are the endpoints of the 3D model line segments. The planes are defined by the two endpoints of a data line segment and the focal point of the camera. Thus, for the 3D-to-2D system, E_{fit} is a normalized function of the residual point-to-plane squared error after the best fit 3D-to-2D pose has been determined.

The omission error is measured in a manner similar to that used by the 2D-to-2D system. However, it is based upon the projection of the updated model after the best fit 3D-to-2D pose is generated. The omission is measured relative to the new 2D projections of the 3D model segments.

Two additional points about the implementation of the 3D-to-2D pose algorithm are worth mentioning. First, for a small percentage of cases, the iterative Quasi-Newton method fails to converge to the optimal 3D pose. To overcome this weakness, we use the Levenberg-Marquardt method suggested by David Lowe [Low91]. For a clear summary of this method see [Pre88], pages 542 – 544. Second, to save computation, Kumar's algorithm has been reformulated in terms of state variables associated with each model-data pair, $s \in S$. The sum of these state variables for all s in a particular correspondence c determines the pose. This allows the contribution from a model-data pair s to be added/subtracted from the current sum if s is added/deleted from the current match. This saves considerable amounts of computation during matching, since it removes the the need to loop over the complete set of pairs in c.

3 Inertial-Descent Local Search

The local search matching algorithm employed is a variation of the steepest-descent strategy described in [Bev90]. We call the algorithm *inertial-descent*, and

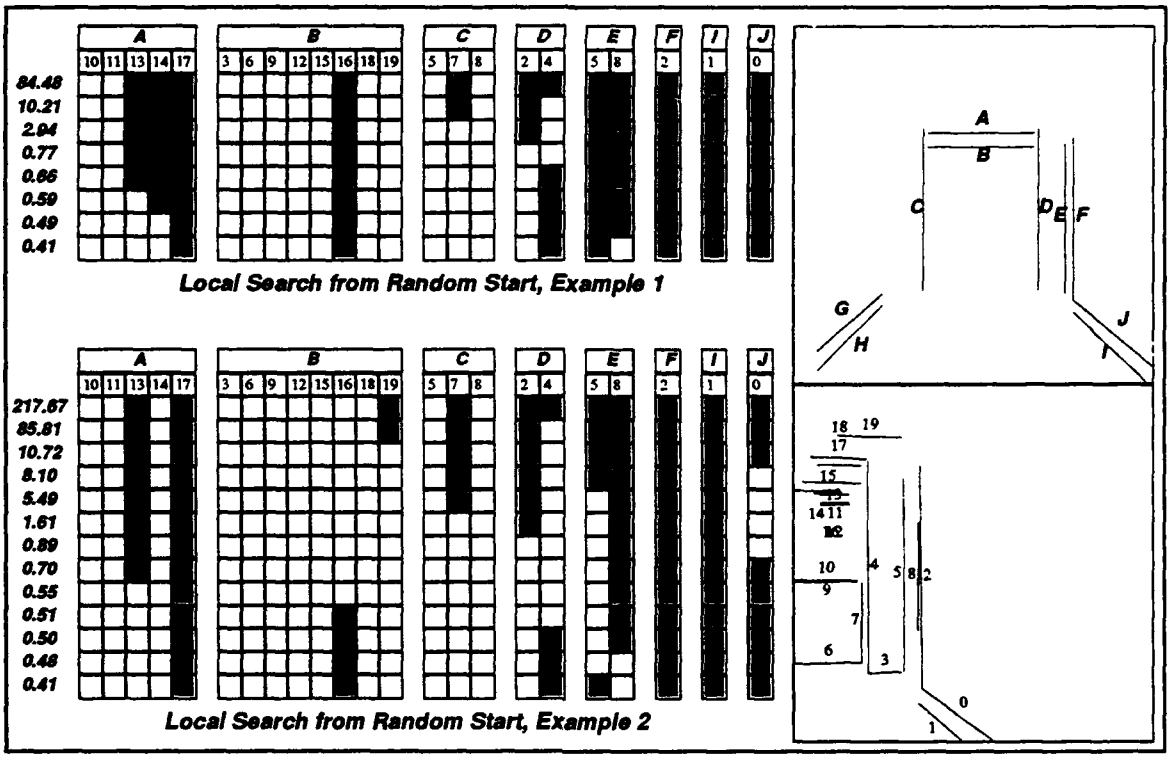


Figure 5: Search space and examples of local search for experiment 2

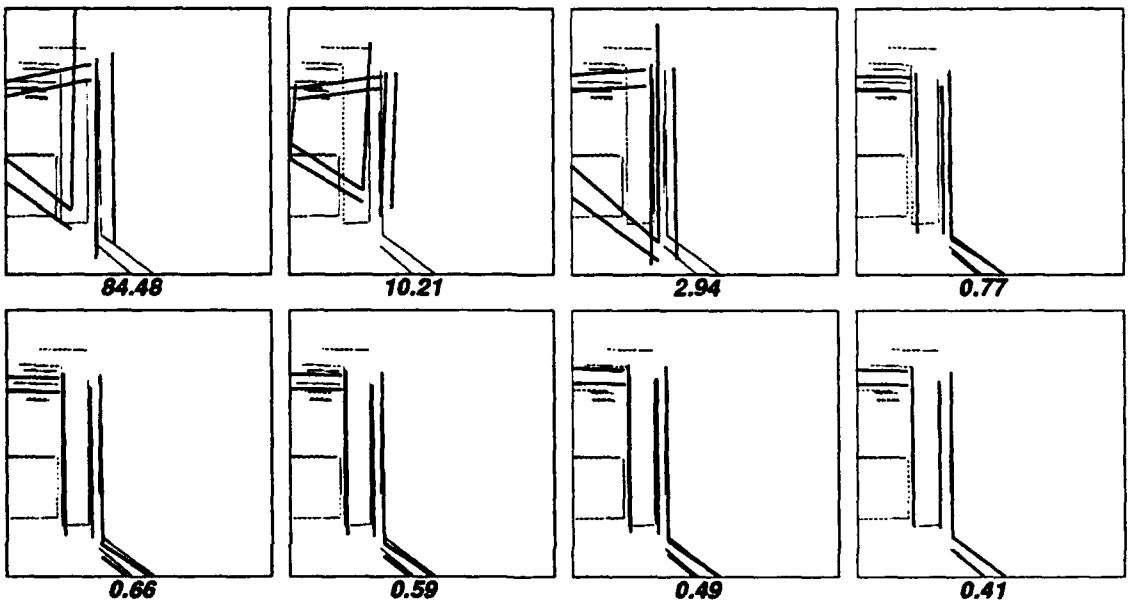


Figure 6: Successive 3D-to-2D pose estimates computed during local search

Match	Improvements in Descending Order
84.48	$((\mathbf{D},4)(D,2)(E,5)(I,1)(J,0)(B,15)(B,12)\ldots$
10.21	$((\mathbf{C},7)(J,0)(I,1)(E,5)(B,12)(B,15)(A,11))$
2.94	$((\mathbf{D},2)(E,5)(E,8)(A,11)(B,12)(B,15))$
0.77	$((\mathbf{D},4)(\mathbf{A},13)(\mathbf{A},14)(\mathbf{E},8)(A,17)(B,15)(B,16))$
0.41	$()$

Table 1: The improvement lists generated by the inertial-descent algorithm each time it tests adding/removing a single pair. The highlighted pairs, applied in sequence, lead to improved matches.

is best explained by example. Figures 5 and 6 illustrate inertial-descent. The matching problem itself is drawn from experiment 2, in which positions A and B are confused (Figure 3). The model and data line segments are shown on the right hand side of Figure 5. The model line segments are shown projected into the image plane as they would appear from position A. The data line segments are derived from the image in Figure 2, acquired at position B . Model line segments are labeled with capital letters, data line segments by number.

Figure 5 also shows a complete trace of two independent runs of the local search algorithm. Each successive row indicates a successively better match. A filled-in square indicates that the associated model-data pair is an element of c. Hence, for example, the first row of the first table indicates a match, c_{init}, selected at random.

$$c_{\text{init}} = \{(A,13),(A,14),(A,17),(B,16),(C,7),(D,2),(D,4),(E,5),(E,8),(F2),(I,1),(J,0)\} \quad (5)$$

The importance of choosing initial matches at random will be discussed shortly. The match error, E_{match}, for each successive match is shown to the left. For the initial match in example 1, $E_{\text{match}}(c_{\text{init}}) = 84.48$.

Each correspondence c may be represented by a bit string. To illustrate, c_{init} from equation 5 may represented as:

$$c_{\text{init}} = 00111\ 00000100\ 010\ 11\ 11\ 1\ 1\ 1 \quad (6)$$

Spacing is maintained to clarify the relation between the columns in Figure 5 and the bits in equation 6.

Toggling a bit in this string adds or removes a feature pair from the current match. A steepest-descent matching algorithm [Bev90] computes E_{match} for all n correspondences which differ from the current c by one bit, i.e. it tests each single bit toggle. It then applies the best single toggle to the current match yielding a new match c'. After each move, steepest-descent revaluates all n possible toggles. It terminates when no toggle yields improvement. In example 1, (Figure 5), the pair yielding the greatest improvement from the initial match c_{init} is $(D,4)$.

Inertial-descent, like steepest-descent, begins by testing all n single bit toggles. Unlike steepest-descent, rather than just applying the single toggle yielding the greatest improvement, inertial-descent builds a sorted list of all toggles yielding improvement. It then works down this list sequentially toggling successive pairs until either: the list is exhausted, or a toggle is found which no longer produces an improvement.

Table 3 illustrates the improvement lists generated by the inertial-descent algorithm for example 1 (Figure 5). The highlighted pairs at the head of the lists actually lead to improved matches. For the first three matches, inertial-descent didn't save any computation relative to steepest-descent. From the new match obtained by toggling the first pair on the list, the second pair no longer improved the match, and therefore all n toggles were again tested. However, for the fourth match, the list allowed the algorithm to apply four toggles in succession without expending effort testing alternatives. In general, inertial-descent tests far fewer matches than steepest-descent.

As already mentioned, each time the local search algorithm tests a new correspondence c, it must compute $E_{\text{match}}(c)$. To emphasis that new 3D poses are generated for each match tested by the 3D-to-2D system, Figure 6 shows the projection of the landmark from these updated poses for the successively better matches found in example 1, Figure 5.

The local search algorithm just illustrated may not find the globally optimal match c^*. However, by running multiple trials from randomly selected initial matches, and then taking the best match found in the series of trials, a local search procedure with a relatively small probability of finding the globally optimal match on a single trial may be used to find the global optimum with very high probability.

Formally, let P_s be the probability of successfully finding the global optimum on a single trial. The conjunctive probability of failing to find the global optimum in t independent trials is Q_f:

$$Q_f = P_f^t \qquad P_f = 1 - P_s \quad (7)$$

Therefore, the number of trials required to find the global optimum with probability Q_s, using a local search algorithm with probability of success P_s, is given by the following equation.

$$t = \lceil \log_{P_f} Q_f \rceil \qquad Q_f = 1 - Q_s \quad (8)$$

To illustrate, if $P_s = 0.10$ then 29 trials are required to find the optimum with probability $Q_s = 0.95$. If $P_s = 0.5$ then only 5 trials are required to reach the same confidence level.

Empirical trials on the landmark recognition problems presented below are used to estimate the true probability of success, P_s, and hence the number trials, t_s, required to obtain the optimal match with confidence 95% or better. In general, 100 trials of local search are used to generate estimates $\hat{P}_s$ and $\hat{t}_s$.

The estimate $\hat{P}_g$ is subject to one important qualification: it may not be based not upon the true globally optimal match c^*, but instead upon what we'll call the optimal match, the best match found in a series of empirical trials. By visual inspection one can often conclude that the optimal match and the globally optimal match are the same. However, this is not always the case.

4 Experiment 1

The task is to recover the true position of the robot from each of the nine pose estimates shown in Figure 3C. [1] The true position, position A, is 41.3 feet from the doorway and 4 feet from each of the two side walls. The nine test positions were obtained by translating the true position estimate forward and backward and side-to-side. Estimates $1-3$ are 5.3 feet forward of position A. Estimates $4-6$ are 1.3 feet forward of A, and estimates $7-9$ are 3.7 feet back of A. Estimates 1, 4 and 7 are 2 feet to the left of A. Estimates 3, 6 and 9 are 2 feet to the right of A. The landmark model as it appears from these nine positions has already been shown in Figure 4.

In addition to varying the initial pose estimates, this experiment considers both 'directed' and 'undirected' model segments. A directed segment specifies the sign of the intensity gradient across the edge. By experimenting with search spaces generated both with and without directed segments we show that using directed segments saves computation. However, at least for the 3D-to-2D system, the quality of the final match and the associated updated pose estimate is the same in each case.

The final pose estimates generated by the 3D-to-2D system are within 0.1 feet of the true pose for all nine initial estimates and for both the directed and undirected search spaces. Essentially the same match is found in all cases, which explains the consistently good final pose estimate. The full 3D perspective approach appears to be very robust.

Figure 8 shows the model and data line segments. The model is projected from the true position. The optimal match, c^*, found by the 3D-to-2D system for the directed search space is:

$$\begin{aligned} c^* \quad = \quad & \{(A,32),(A,33),(B,30),(C,15), \\ & (D,12),(E,14),(F,11),(G,10), \\ & (H,9),(I,7),(J,0)\} \end{aligned} \tag{9}$$

The final pose estimates for the 2D-to-2D system are presented in Figure 7. For cases 2, 5 and 8, for both the directed and undirected search spaces, the 2D-to-2D system recovers the robot's true position essentially as well as the 3D-to-2D system. This is to be expected, since forward and backward error primarily changes the expected scale of the landmark model. However, for the other cases the recovered pose estimates are not as good.

[1] In this experiment we only consider the position portion of the pose estimate associated with a match.

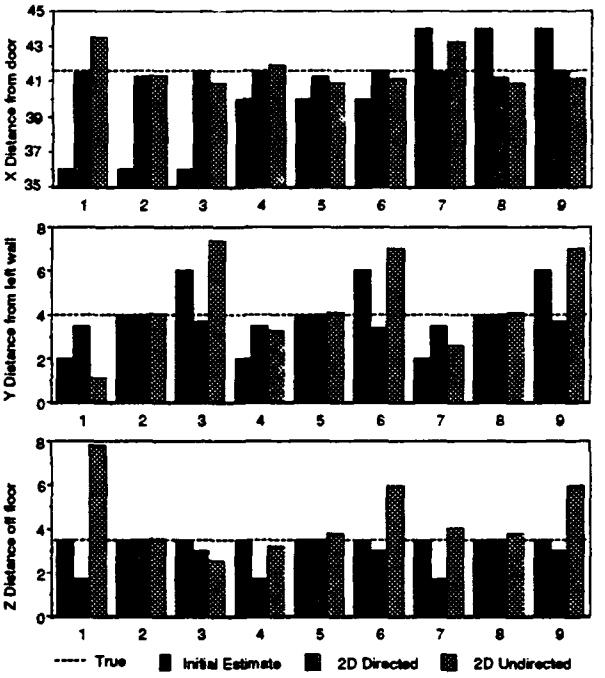


Figure 7: Bar chart of position estimates recovered with 2D-to-2D matching. Position is shown along each of the three dimensions for the best matches. Results for the directed and the undirected search spaces are shown.

For the directed search space the final position is always better than the initial estimate. However, for the undirected search space this is not always the case. In particular, for cases 3, 6 and 9 the recovered Y position is worse than the initial estimate.

The first step in generating these results was to determine the set of candidate pairs S. The selection of candidate pairs depends upon the placement of the 2D projection of a model line m in the image. For the case of directed model segments, a pair $s = (m,d) \in M \times D$ is an element of S_d if:

1 d is within 30 degrees of m.
2 d is within 128 pixels of m.
3 d is at least 1/4 the length of m.
4 d and m have the same sign of contrast.

For the undirected case, S_u, the sign of contrast test is omitted. These bounds are picked based on experience with the domain. In particular, 128 pixels is one quarter the distance across the full 512 by 512 image and is adequate to ensure the correct match is contained in the resultant search space. The number of candidate pairs in S_d and S_u for each of the nine pose estimates are summarized in Table 4.

We tested two alternative ways of generating the initial random correspondences used as starting points by the local search algorithm. In one case, an initial cor-

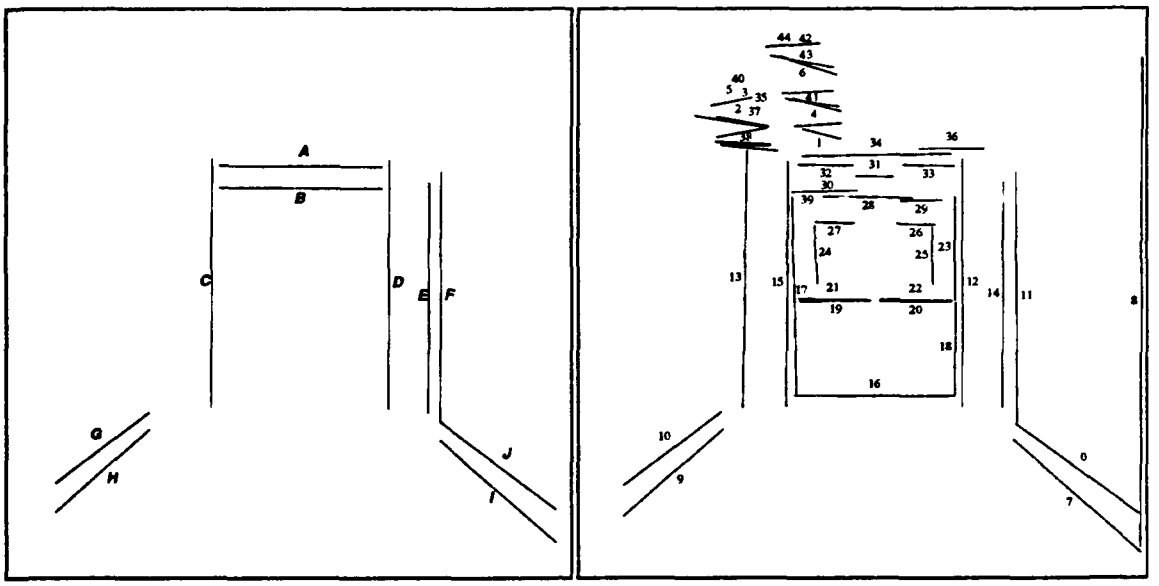


Figure 8: Labeled landmark and data line segments for Image 1

	Initial Pose Estimate								
	1	2	3	4	5	6	7	8	9
$\|S_d\|$	41	36	42	45	37	42	53	43	54
$\|S_u\|$	87	75	89	94	77	94	112	92	112

Table 2: The number of candidate pairs for each of the nine initial pose estimates and using directed, S_d, and undirected, S_u model segments.

respondence c_{init} was selected uniformly from the complete search space C. In the other, we biased the selection to favor choices of c_{init} with roughly 2 data segments bound to each model segment.

This is done by defining a binding probability, $P_B(s)$, such that a pair $s = (m, d)$ is included in an initial correspondence c_{init} with probability $P_B(s)$. Choosing $P_B = 0.5$ for all pairs yields the uniform sampling mentioned above. Defining $P_B(s)$ as follows biases selection. Let k_m be the number of pairs in S which include model feature m, and then define

$$P_B(s) = \max(0.5, 2/k(m)) \qquad (10)$$

The maximum of 0.5 is desirable because it randomizes cases where there are only 1, 2 or 3 candidate pairs for a model segment.

Results comparing uniform randome versus biased random selection are presented in Figure 9. These results are for the directed search space. Both systems were run 100 times for each of the nine position estimates. Figure 9 shows the number of times the optimal match was found in each case. These outcomes suggest the true probability of success, P_s, is higher for biased random selection, and that biased selection is superior to uniform selection. The biased selection defined by equation 10 is therefore used henceforth.

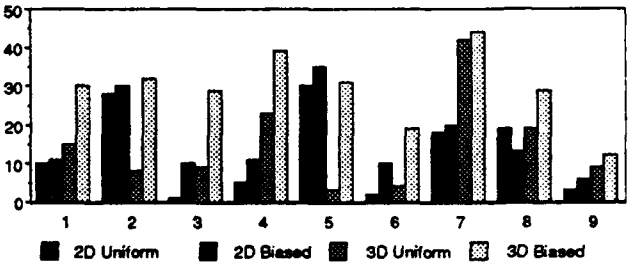


Figure 9: Trials out of 100 yielding the optimal match for uniform verses biased selection of initial feature bindings.

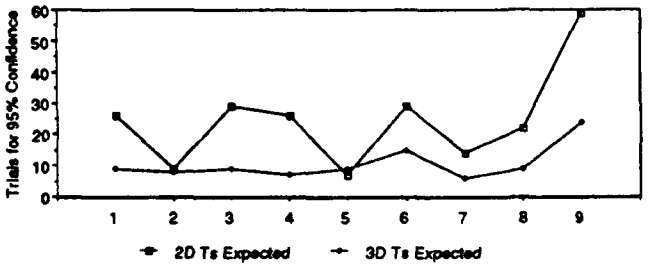


Figure 10: The number of trials required to find the optimal match with 95% confidence

The true probability of success, P_s, is a parameter of a binomial - success/failure - process. The maximum likelihood estimate, $\hat{P}_s$, is just the ratio of the number

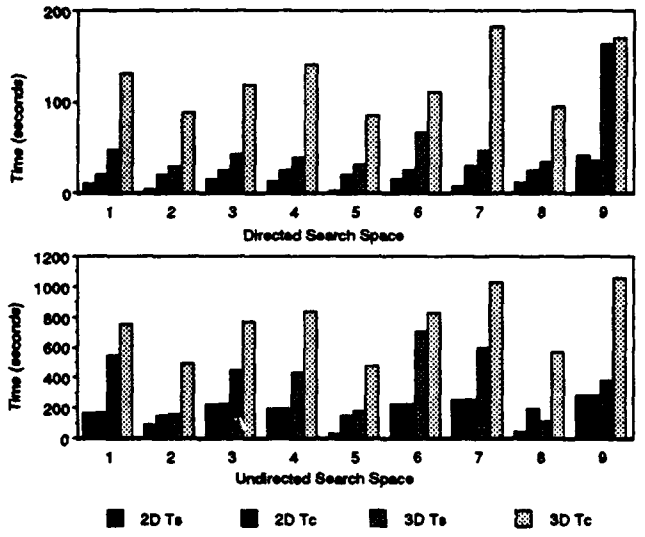


Figure 11: Timing information in seconds for a TI Explorer II Lisp Machine. For both the directed and undirected search spaces the 2D-to-2D system runs in roughly 1/5 the time required by the 3D-to-2D system.

of times the optimal match is found over the total number of trials. Plugging $\hat{P}_s$ into equation 8 yields the estimated number of trials, $\hat{t}_s$, required to find the optimal match with confidence 95% or better. The graph in Figure 10 compares $\hat{t}_s$ for the 3D-to-2D and 2D-to-2D systems over all nine initial pose estimates. These results are for the directed search space. Observe that $\hat{t}_s$ for the 2D-to-2D and 3D-to-2D systems are more similar for cases 2, 5 and 8. Also note the 3D-to-2D system generally requires fewer trials.

The expected amount of time required to find an optimal match is the expected amount of time to run a single trial times the expected number of required trials. Timing information for the 2D-to-2D and 3D-to-2D systems is presented in Figure 11. All times are reported in seconds and are for a TI Explorer II Lisp Machine. Two numbers are given for each case, the first is the number of seconds required to run $\hat{t}_s$ trials. The second number is a more realistic estimate, based upon the time required to run a conservative number of trials, t_c. A conservative number is selected which essentially guarantees finding the optimal match with confidence 95% across every one of the set of problems.

For the directed search space and the 2D-to-2D system, $t_c = 50$ is chosen. Since $\hat{P}_s$ is consistently higher for the 3D-to-2D system, a value of $t_c = 25$ is chosen for this system. For the undirected space the 3D-to-2D system finds the optimal match less often, and $t_c = 50$ is necessary.

For the undirected space and the 2D-to-2D system the definition of success is extended to include the best three matches found. This is necessary because for all but cases 2, 5 and 8 the probability of finding the single uniquely best match becomes very low. Making this change, and running 300 trials, $\hat{P}_s \geq 0.02$. Under these conditions, $t_c = 150$. To be conservative, the worst pose generated by the top three matches was reported in Figure 7.

5 Experiment 2

Consider confusing position A with position B (Figure 3). For this problem, Table 3 shows the true positions, initial pose estimates, and recovered pose estimates. The 3D-to-2D system recovers the true pose to within 1 foot in both cases. However, in recovering position A from an initial estimate of position B, *Est-B-True-A*, the 3D-to-2D system found the best match only once in 300 trials. The next best match was almost equally good, and was found in five out of 300 trials. Success in this case is redefined as finding one of the two best matches, $\hat{P}_s = 0.02$. The recovered pose for the two best 3D-to-2D matches is shown for this case. The 2D-to-2D system did less well, improving the initial estimate in each case, but still missing the true position by several feet.

Est-A-True-B				
	X	Y	Z	Δ
True	30.0	4.0	3.5	
Estimate	40.0	4.0	3.5	10.0
2D	27.0	5.1	2.8	3.3
3D	30.3	3.7	3.9	0.6

Est-B-True-A				
	X	Y	Z	Δ
True	41.3	4.0	3.5	
Estimate	30.0	4.0	3.5	11.3
2D	46.2	1.6	5.6	5.5
3D	40.7	3.8	3.8	0.7
$3D^e$	41.1	3.6	3.9	0.6

Table 3: Pose results when positions A and B are confused. The 3D-to-2D system recovers pose well, but in the *Est-B-True-A case* two almost equally good matches are found, noted as $3D$ and $3D^e$. The 2D-to-2D system does less well.

The estimated probability of success, $\hat{P}_s$, and required number of trials, $\hat{t}_s$, was determined for both systems on both problems. Based upon $\hat{t}_s$, a conservative number of trials t_c was selected, and finally the expected time required to run t_c trials determined. These results are presented in Table 5. As before, times were measured on a TI Explorer II Lisp Machine.

	Est-A-True-B		Est-B-True-A	
	2D-to-2D	3D-to-2D	2D-to-2D	3D-to-2D
$\hat{P}_s$	0.11	0.22	0.08	0.02
$\hat{t}_s$	26	13	36	150
t_c	50	150	50	150
seconds	10	585	35	1,710

Table 4: The estimated number of required trials, $\hat{t}_s$ for Experiment 2. Also the conservative number of trials, t_c and the time required to run this many trials.

6 Conclusion

These experiments provide insight into the importance of perspective. We've compared 2D-to-2D matching with 3D-to-2D matching under conditions where the 2D-to-2D approach might be expected to fail. The results suggest that the 2D-to-2D approach is more useful and reliable than one might at first expect. The results also suggest that the additional cost of doing full 3D-to-2D matching is not prohibitive, and a prudent system might choose to always employ 3D-to-2D matching.

Hybrid algorithms, which blend 2D-to-2D and 3D-to-2D matching, present intriguing possibilities for the future. It is worth noting that the 2D-to-2D system usually improved errorful pose estimates. Hybrid algorithms might well recover 3D-to-2D matches as reliably as the full 3D-to-2D system used here, but with run times closer to those shown for the 2D-to-2D system. One promising hybrid might use 2D-to-2D matching initially and switch to 3D-to-2D matching only after a 2D-to-2D optimal match has been found. Another variation might use the comparatively cheaper 2D-to-2D test to determine a next candidate move, double checking the move by computing a new 3D pose and reprojecting the model.

References

[Bur86] J. B. Burns, A. R. Hanson, and E. M. Riseman. Extracting straight lines. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-8(4):425 - 456, July 1986.

[Bev89] J. Ross Beveridge, Rich Weiss, and Edward M. Riseman. Optimization of 2-dimensional model matching. In *Proceedings: Image Understanding Workshop*, pages 815 - 830, Los Altos, CA, June 1989. DARPA, Morgan Kaufmann Publishers, Inc (Also a Tech. Report).

[Bev90] J. Ross Beveridge, Rich Weiss, and Edward M. Riseman. Combinatorial optimization applied to variable scale 2D model matching. In *Proceedings of the IEEE International Conference on Pattern Recognition 1990, Atlantic City*, pages 18 - 23. IEEE, June 1990.

[Fen90] Claude Fennema, Allen Hanson, Edward Riseman, J. R. Beveridge, and R. Kumar. Model-directed mobile robot navigation. *IEEE Trans. on Syst., Man, Cybern.*, 20(6):1352 - 1369, November/December 1990.

[Kum89] Rakesh Kumar and Allen Hanson. Robust estimation of camera location and orientation from noisy data having outliers. In *Proc. of IEEE Workshop on Interpretation of 3D Scenes*, pages 52 - 60, Austin, TX, 1989. IEEE.

[Kum90] Rakesh Kumar and Allen Hanson. Analysis of different robust methods for pose refinement. In *Proc. of IEEE Workshop on Robust Methods in Computer Vision*, pages 161 - 182, Seattle, WA, 1990. IEEE.

[Ker72] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Tech. Journal*, 49:291 - 307, 1972.

[Lin73] S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498 - 516, 1973.

[Low91] David G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(5):441 - 450, May 1991.

[Pre88] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, Cambridge, 1988.

[Pap82] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*, chapter Local Search, pages 454 - 480. Prentice-Hall, Englewood Cliffs, NJ, 1982.

Subsection E:

Object Recognition and Scene Analysis

Recognition of 3-D Objects from 2-D Groupings *

Fridtjof Stein and Gérard Medioni
Institute for Robotics and Intelligent Systems
Powell Hall 204
University of Southern California
Los Angeles, California 90089-0273
Email: stein@iris.usc.edu

Abstract: We propose an approach for the recognition of three-dimensional objects in a two-dimensional scene. As models, we use a set of non-registered views of a three-dimensional object. By using perceptual organization we develop a hierarchy of features based on proximity, symmetry, parallelism, and closure. The detection of these features is performed in an efficient way using *proximity indexing*. While most other systems use spatial relations for the recognition process, we use high level features and their *topology*. Using indexing, we retrieve matching hypotheses, which are verified against each other with respect to topological constraints. Groups of consistent hypotheses represent detected model instances in a scene. We present results with our current system and discuss further extensions.

1 Introduction

Most object recognition systems today address the problem of finding the location and orientation of an exactly known rigid object in a scene. The tools used to achieve this task are *geometric constraints*, and a lucid treatment of this class of approaches can be found in Grimson's book [7]. The presence of a model is inferred by the verification that such a model could indeed produce some of the observed data under an appropriate geometric transform. It is clear that, with this approach, one can use local, low level primitives such as edgels or their approximations, as produced by state of the art edge finders. Such an approach is therefore very appropriate when evolving in a controlled environment, such as a factory, where the number of possible objects is small and their geometry is precisely known, but cannot be extended to more general scenarios for the following two reasons:

- It is quite difficult to build accurate geometric models (unless they are designed that way).
- When the object library is large, it becomes necessary to develop methods for indexing into the library to select likely objects.

One possible way to circumvent these problems is to use projective invariants [6], but these must be hand selected.

We propose instead that the solution to these problems lie in *groupings* of the initial primitives. Such groupings serve as an intermediate level representation of the data, in a hierarchical fashion, and can be used to retrieve likely candidate objects from a library. Furthermore, these can be extracted from multiple views of an object, rather than from a complete model (as such they are "quasi-invariants" [1, 14]). This idea of "perceptual grouping" is hardly new. It can be found in the psychology literature of the 1920s, under the *Gestalt* name. As an explanation of how the perception of individual objects are formed, the Gestalt theory proposes an organization of parts of an image into wholes, based on laws of grouping. Elements in the image are grouped based on proximity, similarity, closure, symmetry, and continuation. These groups themselves can be used as elements and grouped with the same laws. Unfortunately, the implementation of such observations is difficult, as theses laws conflict, even for simple stimuli. In computer vision these ideas can be found in [13] and [24].

The issues we have to tackle in order to generate groupings relate to the choice of such groupings, their discrimination power, their robustness to viewpoint and noise, and their efficient computation.

Here, we address these issues and develop a feature hierarchy which can be used for object recognition of three-dimensional objects from a two-dimensional scene. In this hierarchy, we propose specific groupings based on proximity, parallelism, symmetry, and closure. The detection of these features is performed in an efficient way using *proximity indexing*. First, we generate features with multiple representations to overcome the unreliability of local algorithms during preprocessing, and to handle noise and capture different levels of detail. Later, we merge perceptual similar features at higher levels of the feature hierarchy. As models, we use a set of non-registered views of a 3-D object. While most other systems use spatial correspondences to verify matching hypotheses, we use high level features and their *topological* relationships for the recognition process. These features are grouped based on closure and proximity to generate so called *high level groupings* which are stored in a table. Using indexing, we retrieve matching hypotheses, which are verified against each other with respect to topological constraints. Groups of consistent hypotheses represent detected model instances in a scene.

Our proposed groupings are not guaranteed to produce "natural" primitives, in the sense of the axis of a generalized cone for instance, so we need to validate our choice. This is achieved by demonstrating a recognition system which, even in its early implementation, can recognize an object, known from a significantly different viewpoint, and using topologic constraints as opposed to geometric constraints.

Our paper is organized as follows: we start with a brief review of previous work in Section 2. Section 3 describes the feature hierarchy, discussing the detection algorithms which we developed to derive the different features from the intensity edges of an image. We then introduce *Proximity Indexing* in Section 4. In Section 5

*This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-90-C-0078 and by a NSF Grant under award No. IRI-9024369 .

we talk about the motivation of using high level groupings and give an outline of an object recognition system which is in its early stage, and we show some promising preliminary results.

2 Previous Work

As most of the previous work is concerned with pose estimation rather than object recognition, we simply refer the reader to [7] for an excellent overview. Other approaches which focus on groups of low level primitives combined with an indexing scheme for the purpose of two-dimensional object recognition can be found in [2, 4, 12, 10]. The method described in [22] is concerned with three-dimensional recognition. Using explicitly groupings for recognition was done by the following authors:

Lowe [13] gives an excellent introduction to the ideas behind perceptual grouping, significance of features, and an implementation of a system called SCERPO which incorporates some of his ideas. SCERPO is an recognition system which is able to estimate the pose of a three-dimensional CAD model in a two dimensional scene based on line segments.

Mohan and Nevatia [15] perform segmentation of images based on perceptual clues. They use the resulting segmentation for the detection of buildings in aerial images and for solving the stereo correspondence problem. Their groupings are strongly *natural* in the sense, that the segmentation corresponds to a physical interpretation. A special focus on *natural* descriptions of objects was also done by Rao and Nevatia [17].

Huttenlocher and Wayner [9] describe an algorithm to find the largest convex groupings of line segments. The advantage of such an approach is that the number of groups generated is smaller than the number of original primitives. The role of such groupings in recognition has yet to be demonstrated, however.

A different approach of perceptual grouping is taken by Sha'ashua and Ullman [20]. They focus on the saliency of structure in an image. They present a saliency measure based on curvature and curvature variation, and they show examples of cluttered scenes where their results correspond to the human focusing mechanism.

3 Going from Edgels to Groupings

We now explain the steps involved in going from an image to a high level representation of it in terms of "perceptual" groups. This chain of processing is sketched in Figure 1. We start with an image. In the *preprocessing* stage we reduce the amount of data: Starting from images we compute curves, which consist of linked edgels.

In the *local grouping* stage we generate many line segments based on multiple linear approximations with different fitting tolerances. For each approximation tolerance, we perform a vertex collapse and compute super segments and parallels with exhaustive grouping. By creating a large set of features at this point we gain robustness in our further groupings, and we significantly reduce the unreliability of the preprocessing.

The *perceptual grouping* stage no longer distinguishes between features of different fitting tolerances. The reduction of data is based on two strategies:

1. merging perceptual similar features from the local grouping stage, and
2. grouping features into higher level features using geometric relationships such as symmetry, closure, and proximity.

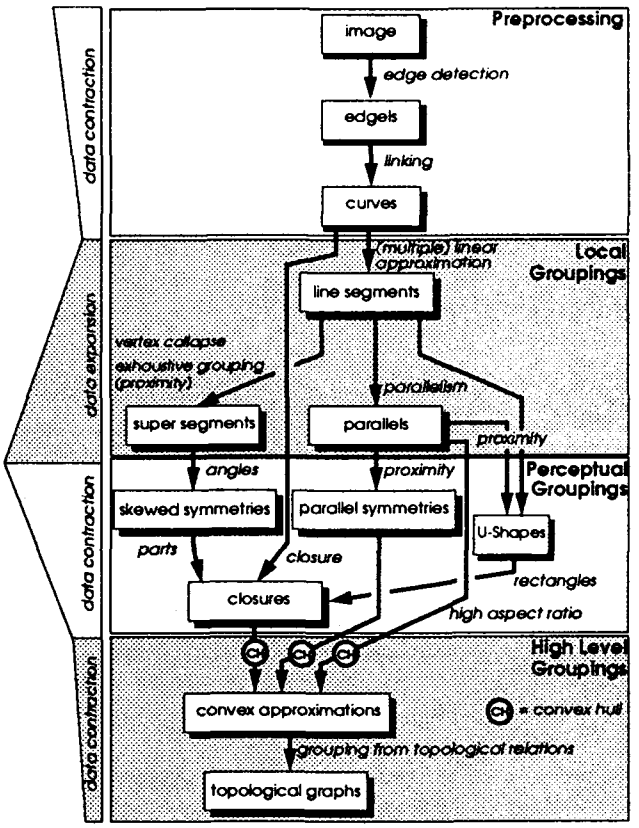


Figure 1: Hierarchy

To deal with all the results from the previous stage, we use a coarse approximation in the *high level grouping*, namely the convex hull. With the convex approximations we build a topological adjacency graph based on common segments or vertices. High level features consist of groupings of closure and proximity in this adjacency graph.

3.1 Preprocessing

We first apply an edge detection algorithm on the image (we use the Canny edge detector [3]). The resulting edgels are further linked into *curves*. Curves are then approximated with a line fitting algorithm to compute the polygonal approximations. Instead of just using one representation we approximate each curve with a set of line fitting tolerances to get a robust representation. For every approximation, we then collapse vertices, so that vertices which are close together (typically three to five pixels) are collapsed into one vertex.

3.2 Super Segments

Idea Since we want to handle occlusion, we do not expect to obtain complete boundaries in our images, but only portions of them. On the other hand, individual segments are too local to be useful as matching primitives. Grouping a fixed number of adjacent segments provides us with one of our basic features, the super segments.

Implementation The computation of super segments is the same as described in [21]. Connected linear segments form chains of adjacent segments. The segment

chains provide the super segments by grouping a fixed number of adjacent segments. (Because of the branching of the polygonal approximations we generate super segments exhaustively by generating them from all possible segment combinations.) We typically generate super segments of cardinality three to six.

3.3 Parallels

Idea A *parallel* consists of two linear segments s_1 and s_2. We require that the two segments overlap, which means that the normal projection of s_1 on s_2, or the normal projection of s_2 on s_1 is not empty. Furthermore we do not want the aspect ratio of the parallel to reflect elongation, with $\text{ar}(p) = (\text{length}(s_1) + \text{length}(s_2))/\text{distance}(s_1, s_2)$. Typically we require $\text{ar}(p) > 0.5$.

Implementation The acquisition of the parallels is performed in two steps by using proximity indexing (for details, see Section 4).

1. All segments are recorded using the quantized orientation as the key. We use δ as quantization (typically $\delta = 20°$). Using proximity indexing we are guaranteed to find parallels which are at most $\delta/2$ apart and we get some parallels with an enclosed angle between $\delta/2$ and δ.
2. For every linear segment, the possible candidate parallels are retrieved and verified with respect to aspect ratio and overlap. Segment pairs which meet theses constraints generate parallels.

3.4 Symmetries

A symmetry is defined as a one-to-one mapping between the points of two curves, with the symmetry axis defined as the locus of the mid-point of the straight lines joining a point on one curve to its image in the other [15]. Ulupinar and Nevatia have proposed two specific symmetries, namely skewed and parallel symmetries [23]. Whereas they use them to infer surface orientation, we consider them as a general feature of an object.

3.4.1 Parallel Symmetries

Idea Given two curves $X_i(s) = (x_i(s), y_i(s))$, for $i = 1, 2$, parametrized by arc length s, and $\theta_i(s) = \arctan((dy_i(s)/ds)/(dx_i(s)/ds))$. $X_1(s)$ and $X_2(s)$ are said to be parallel symmetric [23] if there exists a point-wise correspondence $f(s)$ between them such that $\theta_1(s) = \theta_2(f(s))$ for all values of s for which X_1 and X_2 are defined, and $f(s)$ is a continuous monotonic function. We only consider the special case where $f(s)$ is a linear function.

Implementation Parallel symmetries are retrieved by finding proximate parallels. We do not use the super segment approach, because we would depend on the cardinality of the super segments. By using the parallels as the building blocks, we can use proximity indexing to find parallels which share the same vertices. The acquisition of the parallel symmetries is performed in two steps.

1. We record every parallel twice. One time with the sorted list of the vertex coordinates of one side as key, the other time with the sorted list of the vertex coordinates of the other side as key.
2. For every parallel, the possible neighbor parallels are retrieved. The groups of adjacent parallels generate parallel symmetries.

One example is shown in Figure 2.

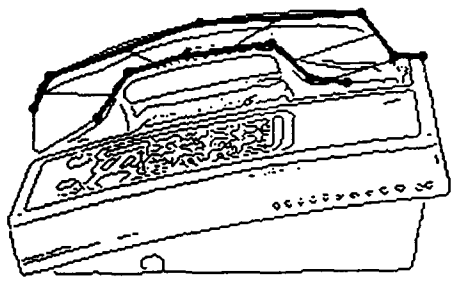


Figure 2: Example of a Parallel Symmetry

3.4.2 Skewed Symmetries

Idea In a skewed symmetry, the point-wise correspondence is such that the axis of the symmetry is straight, and the lines of symmetry are at a constant angle (not necessarily orthogonal) to the axis of symmetry. Skew symmetry was first proposed by Kanade [11] and used in the analysis of scenes of polyhedral objects. An example is given in Figure 3(a).

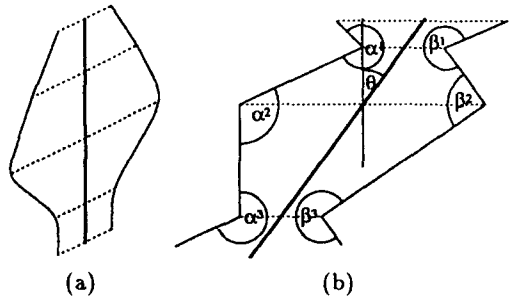


Figure 3: Examples to (a) skew symmetry with curved contours, (b) and skew symmetry with straight contours. The bold curves are axis of symmetry and the dotted lines are lines of symmetry.

The detection of skew symmetry for curves was done by Ponce [16] and Saint-Marc and Medioni [19], but these methods are quite sensitive to noise.

In our system, we are interested in symmetries between line segments. Considering all possible symmetries between line segments as proposed by [8] is expensive.

Our approach is based on finding skew symmetries between super segments. An example of skew symmetry for straight line segments is given in Figure 3(b). As we show in the Appendix, we have $\Delta = |\alpha^i - \beta^i| \leq 2\theta$, where θ is the skew angle. This allows us to define a local signature for a super segment, namely the angles, with which we can find possible symmetry candidates by indexing. This avoids the expensive comparison of all pairwise super segments. To test whether two super segments are skew symmetric, we have to test the following:

1. The difference between the corresponding angles must be smaller than $2\theta_{\max}$.
2. The symmetry axis has to be straight.

In using proximity indexing we get an efficient algorithm.

Implementation Skewed symmetries can be retrieved by finding pairs of super segments with angles of opposite sign. The detection of the skewed symmetries is performed in three steps:

1. We record every super segment using the list of the curvature angles as a key. To allow the guaranteed detection of symmetries with up to skew angle θ_{max}, we choose as the quantization interval for the super segment angles $2\theta_{max}$.

2. For every super segment, we use the list of opposite sign curvature angles to retrieve possible symmetry candidates.

3. Super segment pairs which generate symmetry hypotheses have to be checked to see whether the corresponding symmetry axis is straight. We test this by requiring the scatter matrix of the middle points of all corresponding vertices to have a high eccentricity (they lie all along a line).

We show an example for a set of detected symmetries in Figure 4.

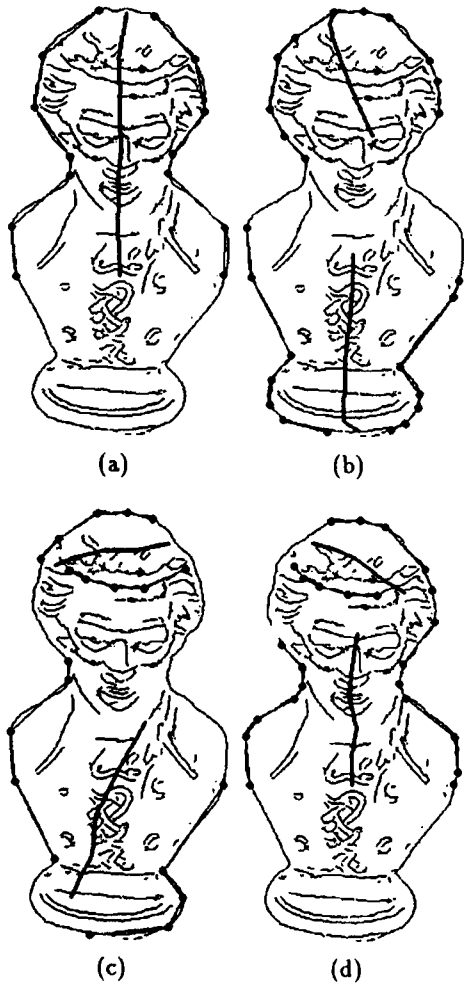


Figure 4: Detected Symmetries and the Corresponding Axes in Mozart Bust

3.5 Closures

Lowe [13] makes the following statement: *There is a tendency for curves to be completed so that they form enclosed regions.* Based on this statement, Mohan and Nevatia [15] developed the idea to close symmetries at their ends to obtain so called *ribbons*, which form enclosed regions. They use these ribbons to segment images.

We want to use closures as features, which do not necessarily have any physical interpretation in the image. At the moment we compute closures from U-Shapes, from closed curves and from skewed symmetries.

3.5.1 Closure from U-Shape

Idea A parallel which is closed at one side by a linear segment is a strong indication that a rectangular structure is at hand where one side could not be detected. We therefore assume that we found a closed contour.

Implementation U-Shapes can be found by indexing over the vertex pairs of parallels and trying to find a segment which forms a U-Shape with the parallel.

1. We record every parallel twice using once the quantized vertices of one "side" of the parallel and then other vertices as keys, where we record the parallel.
2. For every linear segment we use the list of the endpoints to retrieve possible U-Shape candidates.
3. If the angle between the parallel and the segment is $90° \pm 30°$ we generate a new U-Shape.

3.5.2 Closure from Curve

The obvious form of a closure occurs if we have a closed curve. To detect a closure based on a curve we allow the gap between start and end of the curve to be 5% of the arc length of the curve.

3.5.3 Closure from Skewed Symmetry

Idea We adopt the idea that a segmentation into parts should be done at negative minima of curvature from Rom and Medioni [18]. Such "a part" is used in our system as a closure.

Implementation For every skewed symmetry we traverse the angles of one of its super segments (the other super segment just has the skewed mirror angles). Whenever we encounter a sign change of consecutive angles, we "break" the symmetry at this point and define the symmetry up to this vertex (together with the corresponding vertex of the other super segment) as one part. Applying this step iteratively, we generate alternating convex and concave parts. We use the convex parts to create closures. An example can be seen in Figure 5.

4 Proximity Indexing

We encounter two questions in dealing with the generation of perceptual groups:

1. How do we cluster initial features based on their characteristics and geometric relationship in an efficient way, in order to get a combination which represent a *higher level* features?
2. How do we merge perceptual features to avoid multiple features with the same perceptual content?

Both question point towards a a difference measure. Given a feature, we define its characteristics as a *vector* of attribute values (e.g. eccentricity, orientation, location). The difference between two features is then defined as the pairwise difference between the attribute values. If

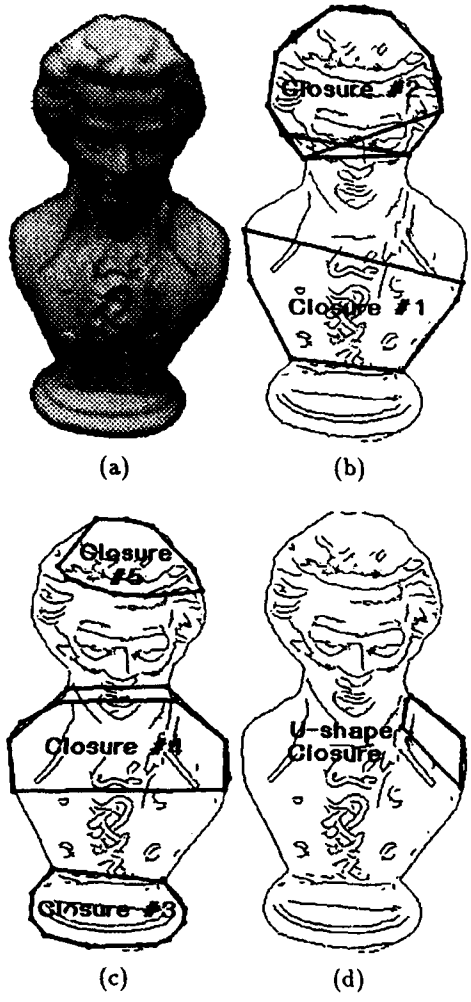


Figure 5: Detected Closures in Mozart Bust: (a) shows the original image, (b) and (c) show five different closures from skewed symmetries, and (d) shows a closure from U-shapes, which is a strong hint of a rectangular structure. We only display the convex hulls.

all the differences are small, then the two features are similar.

Given a set of features, how do we find the ones with similar attribute values? Traditional search methods which compare every possible pair are very time consuming. Therefore recent vision systems [12, 21, 22] have used indexing to find corresponding features with similar characteristics. The big issue in indexing is the crucial parameter for the quantization size and the question: "What happens to values which are close, but fall in different quantization intervals?" Two features match only in the case when both keys are exactly the same. This means that all the pairwise values have to fall into the same quantization intervals. Suppose the range is quantized into intervals of the the same size q. Each value, v, is then assigned a key based on which interval v falls into, and v is corrupted by a random additive term bounded by ε; then the probability that a corrupted vector is assigned the same entry as the original one is [22]

$$p^n(k|v) = (1 - (\varepsilon/q)^n.$$

Looking at this equation, it is obvious that the probability of matching long keys decreases rapidly.

As Flynn and Jain [5] point out, it is essential to have an indexing scheme that preserves proximity in the key values. So far, two strategies based on indexing have been used to deal with this problem: large bucket size and searching of neighboring bins. While large bucket size is based on the hope that "less values will fall into the incorrect bin", the search of neighboring bins has an exponential complexity with respect to the number of false value matches.

We propose an alternative approach: Instead of using all features with all quantizations as a large alphabet, we break the features apart and use indexing on every value separately.

Encoding: To assure proximity we store the feature for every value v twice:

1. under the key $[\lfloor v \rfloor_q, \lceil v \rceil_q]$
2. under the key
$$\begin{cases} [\lfloor v \rfloor_q^+, \lceil v \rceil_q^+] & \text{if } v - \lfloor v \rfloor_q > \lceil v \rceil_q - v \\ [\lfloor v \rfloor_q^-, \lceil v \rceil_q^-] & \text{otherwise,} \end{cases}$$

with

$$\begin{array}{llll} \lfloor v \rfloor_q = q\lfloor \frac{v}{q} \rfloor & & \lceil v \rceil_q = \lfloor v \rfloor_q + q \\ \lfloor v \rfloor_q^+ = \lfloor v \rfloor_q + \frac{q}{2} & & \lfloor v \rfloor_q^- = \lfloor v \rfloor_q - \frac{q}{2} \\ \lceil v \rceil_q^+ = \lceil v \rceil_q + \frac{q}{2} & & \lceil v \rceil_q^- = \lceil v \rceil_q - \frac{q}{2}. \end{array}$$

Retrieval: The retrieval of a feature f is broken up into the retrieval of every value. Every value is quantized twice (see above) and the stored features for both intervals are retrieved and combined into one set. For all values we get such a feature set. The intersection of *all these sets* results in the features which are close to f in *all* their values. Due to the interlaced quantization we can guarantee to retrieve all features with $|v_{\text{stored}} - v_f| < \frac{q}{2}$, and we get some feature matches with $\frac{q}{2} < |v_{\text{stored}} - v_f| < q$. The intersection process can be sped up by intersecting the sets in increasing cardinality.

Complexity: What do we pay for using proximity indexing compared to traditional indexing? Let n be the number of features to store. We assume that every feature has as a key an attribute vector of fixed length. Every value is quantized in a fixed number of intervals. Furthermore we assume that the features are equally distributed over the table which consists of r records. The cost of using proximity indexing is summarized in the following table:

	search	indexing	proximity indexing
preserves proximity	yes	no	yes
encoding complexity	0	$O(n)$	$O(n)^*$ / $O(n \log(\frac{n}{r}))^{**}$
retrieval complexity	$O(n^2)$	$O(n)$	$O(\frac{n^2}{r}\log(\frac{n}{r}))^*$ / $O(\frac{n^2}{r})^{**}$

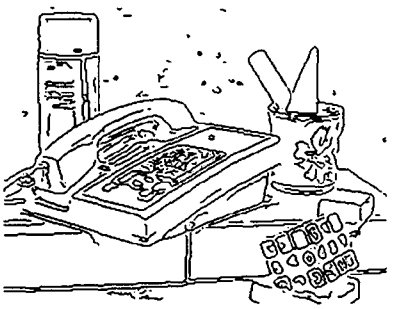
* : entries are unsorted ** : entries are sorted

All tables mentioned in this Section are implemented as hash tables.

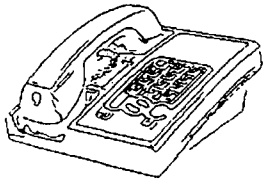
5 Recognition

So far we described the clustering and grouping of very basic features to higher level groupings. But what are the advantages of going higher in the feature hierarchy? As we mentioned above, we want to represent a 3-D

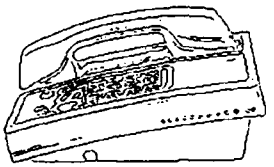
model with multiple 2-D views. But how many views do we need? Making quantitative statements about the number of views which we require for an object is not possible. A qualitative statement can be illustrated with an example. In Figure 6(a) we show a cluttered scene, in



(a) Scene



(b) View #1



(c) View #2

Figure 6: Example Phone

which we want to detect the phone. Figure 6 shows two different views of the phone. In Figure 6(b) the view is very similar to the view in the scene. Figure 6(c) shows a side view of the phone. This corresponds to a rotation of approximately 70° from the view in the scene. By using different features for solving the correspondence problem between the scene and the phone we can show the performance of different features in the feature hierarchy. In the following, we show two approaches:

1. We discuss an attempt to match two views with primitives based on local groupings.
2. We propose an object recognition system which is based on high level perceptual groupings (see Figure 1).

We analyze both approaches and point out the advantages of our proposed feature hierarchy.

Due to the fact that different models and different views mean the same for our object recognition system we will use both terms interchangeably in the following sections.

5.1 Correspondence based on Super Segments

The first example of finding the correspondence with super segments is based on an object recognition system which was designed for the recognition of multiple flat objects [21]. Super segments are computed from the linked image edges and provide the essential mechanism for indexing and retrieval. We use the quantized angles and the eccentricity as a key and record the super segment in a table. The recognition proceeds by computing the super segments of the scene. Each scene super segment retrieves model hypotheses from the hash table. Hypotheses are clustered if they are mutually consistent in their geometric relationship, and represent the instance of a model. This methodology allows us to recognize 2-D models from a 2-D scene in the presence of noise, occlusion, scale, rotation, translation and weak perspective.

Consistency between matching hypotheses require stable geometric relationships. It is obvious that these geometric relations between parts of the image are only valid in similar views. The projection of the view #1 onto the scene is shown in Figure 7.

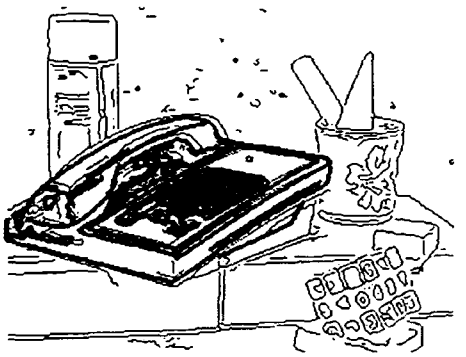


Figure 7: Recognition based on Super Segments

Analysis : The matching presented here succeeds only because the 2-D views of the models are quite similar. In fact, the method fails to identify view #2 in the image. We therefore need a large number of views (a few hundred for a full model!) for each object. This would be a very cumbersome and expensive (space and time) approach! Furthermore, we encounter a rapid growth in the number of hypotheses which have to be verified, when we increase the number of models in the data base. Due to the fact that the verification has a time complexity of $O(n^2)$ (with n the number of generated hypotheses), the recognition process would become unacceptably slow.

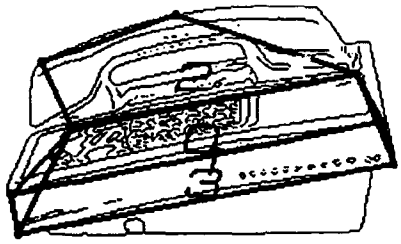
5.2 Correspondence based on High Level Features

Here instead we use high level features to find the correspondence. Instead of using *geometric* relationships between features we exploit *topologic* relationships, such as adjacency, proximity, intersection, etc. These relations between features are more stable with respect to the 3-D viewpoint than properties based on 2-D geometry.

The high level features are generated by using features from the perceptual grouping stage (see Figure 1). Each such feature is approximated by its convex hull. These so called *convex approximations* (CA) are grouped based on closure and proximity: two features are defined as neighbors if they share a common line segment (or common vertex). To find "high level closures" we construct an adjacency graph. In this adjacency graph we look for cycles which represent a closure between CAs.

These groupings can be encoded based on attributes such as the connectivity of the underlying CAs, the two-dimensional topologic relationships between them, and the number of corners of the underlying CAs. Using the code as a key, we record the grouping in a table. A scene is processed in the same way, and by using the resulting high level scene groupings we are able to retrieve corresponding groupings from the table.

In the example in Figure 8 we show such a grouping which consists of four CAs, namely a cycle of CAs (labeled 0,1,3) and another CA (label 2) which is adjacent



(a) Model



(b) Scene

Figure 8: *Recognition based on High Level Groupings*

to CA 0 and CA 1. By allowing a certain tolerance in the number of corners, we can match these two groupings, despite the fact that CA 2 has four corners in the model and five corners in the scene. The recognition of the view #2 in Figure 6(c) in the scene is shown in Figure 8(b). Instead of the projection of the phone itself (it would look too warped) we show the corresponding groupings.

By using groupings which include no geometric relationships in the matching process, we can find correspondences between very different views. This implies that, for the purpose of object recognition, we need much fewer two-dimensional views to represent a three-dimensional model than in the approach discussed in Section 5.1. Furthermore, we claim that such high level features are highly discriminative. To firmly establish that is part of the ongoing research.

6 Conclusions

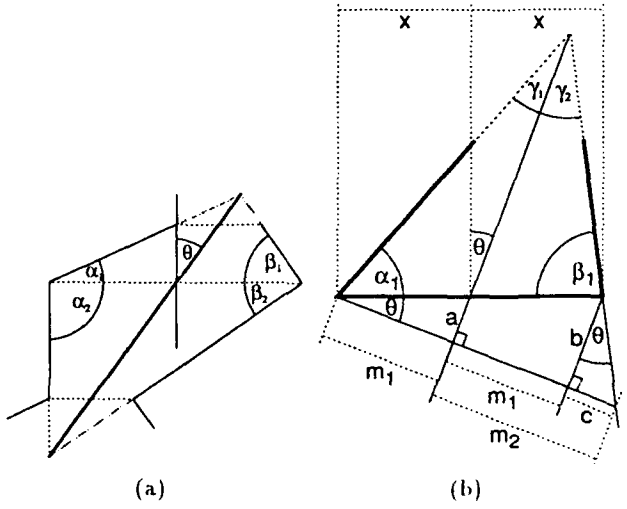
In this paper, we have developed an approach to use perceptual organization for the purpose of object recognition, and show some promising results. The major distinguishing aspects of our system compared to previous approaches are:

- For our object description we use *multiple representations.*
- We *merge* intermediate results to increase the overall reliability of feature detection.
- We *expand and contract* the amount of features.
- Proximity Indexing and the use of linear segments leads to an *efficient* implementation.
- We implemented a first stage of using *high level grouping.*
- *No Early Commitment.* Our perceptual grouping is purely data driven. We do not try to resolve any ambiguities and the groupings do not necessarily lead to a single physical interpretation. By using the feature groups for recognition purposes we make effective use of the underlying organization.
- Our system deals with *topological* relations, not with *spatial correspondences.*
- By using a set of different views to represent a model we can deal with *incomplete model descriptions.*

Our future work aims at answering the following questions:

- What happens when the system does not find corresponding high level groupings (e.g. due to heavy occlusion)? We want to focus on this point by developing a multilevel matching, which allows the system to "fall back" on lower level features in order to find correspondences.
- We want to extend the feature hierarchy by including perceptual organization between high level features. So far we use only proximity and closure to generate high level groupings. We are investigating the possibility of including symmetry and parallelism.
- There are several other features and grouping strategies which we ignore so far: continuation, texture, saliency ... Including these features would enrich the descriptive and discriminative power of our feature hierarchy.

Our preliminary work and the corresponding results in this paper have demonstrated the viability of this approach.



(a) (b)

Figure 9:

Appendix

We claim that

$$\Delta = |\alpha^i - \beta^i| \leq 2\theta. \tag{1}$$

For the sake of simplicity let $\alpha^i = \alpha$ and $\beta^i = \beta$, then (1) becomes

$$\begin{aligned} \Delta &= |\alpha - \beta| = |\alpha_1 + \alpha_2 - \beta_1 - \beta_2| \\ &= |(\alpha_1 - \beta_1) + (\alpha_2 - \beta_2)| \leq 2\theta \end{aligned} \tag{2}$$

To prove (2), we show that

$$-2\theta \leq (\alpha_1 - \beta_1) \leq 0 \tag{3}$$

and

$$0 \leq (\alpha_2 - \beta_2) \leq 2\theta. \tag{4}$$

First let us focus on equation (3). We know from Figure 9(b) that

$$\alpha_1 + \pi/2 + \theta + \gamma_1 = \pi \tag{5}$$

and

$$\beta_1 + \pi/2 - \theta + \gamma_2 = \pi. \tag{6}$$

Subtracting (5) from (6) results in

$$(\beta_1 - \alpha_1) + (\gamma_2 - \gamma_1) = 2\theta. \tag{7}$$

Furthermore we know that

$$\gamma_2 + \gamma_1 \leq \pi. \tag{8}$$

By showing that $\gamma_2 \geq \gamma_1$ we can show that (3) is true. The proof of $\gamma_2 \geq \gamma_1$ is divided into two cases:

1. $\gamma_2 \geq \pi/2$: Considering equation (8) we get $\gamma_2 \geq \gamma_1$.
2. $\gamma_2 < \pi/2$: Based on projective geometry we get

   $$\gamma_2 \geq \gamma_1 \Leftrightarrow m_2 \geq m_1$$

   with $m_2 = m_1 + c$. If we can show that $c \geq 0$, we show that $\gamma_2 \geq \gamma_1$. We know that

   $$c = b \tan\theta. \tag{9}$$

   Furthermore we get from the projective geometry a relation between a and b:

   $$a/x = b/2x \Leftrightarrow b = 2a. \tag{10}$$

   Since $a = x \sin\theta$ we get from equation (9) and (10) $c = 2x \sin\theta \tan\theta$, and therefore $c \geq 0$ for $0 \leq \theta \leq \pi/2$.

Therefore equation (3) is true. Equation (4) can be shown in the same way.

Q.E.D.

References

[1] T.O. Binford. Inferring surfaces from images. *Artificial Intelligence*, 17:205–244, 1981.

[2] T.M. Breuel. Adaptive model base indexing. In *Proceedings of the DARPA Image Understanding Workshop*, pages 805–814, 1989.

[3] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[4] G. J. Ettinger. Large Hierarchical Object Recognition Using Libraries of Parameterized Model Sub-parts. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, Ann Arbor, Michigan, June 1988.

[5] P. J. Flynn and A. K. Jain. 3d object recognition using invariant feature indexing of interpretation tables. In *IEEE Workshop on Directions in Automated CAD-Based Vision*, pages 115–123, Maui, Hawaii, June 1991.

[6] D. Forsyth, J. L. Mundy, A. Zisserman, and C. M. Brown. Invariance - a new framework for vision. In *Proceedings of IEEE International Conference on Computer Vision*, pages 598–605, Osaka, Japan, December 1990.

[7] W. E. L. Grimson. *Object Recognition by Computer-The Role of Geometric Constraints*. MIT Press, Cambridge, MA, 1990.

[8] L. Herault, R. Horaud, F. Veillon, and J.J. Niez. Symbolic Image Matching by Simulated Annealing. In *Proceedings of the British Machine Vision Conference*, pages 319–324, University of Oxford, England, September 1990.

[9] D. P. Huttenlocher and P. C. Wayner. Finding Convex Edge Groupings in an Image. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 406–412, Maui, Hawaii, June 1991.

[10] A. Kalvin, E. Schonberg, J. T. Schwartz, and M. Sharir. Two-Dimensional, Model-Based, Boundary Matching Using Footprints. *International Journal of Robotics Research*, 5(4):38–55, 1986.

[11] T. Kanade. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17:409–460, 1981.

[12] Y. Lamdan and H. J. Wolfson. Geometric Hashing: A General and Efficient Model-Based Recognition Scheme. In *Proceedings of IEEE International Conference on Computer Vision*, pages 218–249, Tampa, Florida, December 1988.

[13] D. G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Hingham, MA, 1985.

[14] D.G. Lowe and T.O Binford. Perceptual organization as a basis for visual recognition. In *Proceedings of AAAI-83*, Washington, D.C., August 1983.

[15] R. Mohan and R. Nevatia. Using perceptual organization to extract 3-d structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1121–1139, November 1989.

[16] J. Ponce. On characterizing ribbons and finding skewed symmetries. *Computer Vision, Graphics and Image Processing*, 52:328–340, 1990.

[17] K. Rao. *Shape Description from Sparse and Imperfect Data*. PhD thesis, University of Southern California, December 1988. IRIS Technical Report 250.

[18] H. Rom and G. Medioni. Hierarchical decomposition and axial representation of shape. In *Proceedings of the DARPA Image Understanding Workshop*, San-Diego, California, January 1992. (these Proceedings).

[19] P. Saint-Marc and G. Medioni. B-spline contour representation and symmetry detection. In *First European Conference on Computer Vision*, pages 604–606, Antibes, France, April 1990.

[20] A. Sha'ashua and S. Ullman. Structural saliency: the detection of globally salient structures using a locally connected network. In *Proceedings of IEEE International Conference on Computer Vision*, pages 321–327, 1988.

[21] F. Stein and G. Medioni. Efficient Two Dimensional Object Recognition. In *Proceedings of International Conference on Pattern Recognition*, Atlantic City, New Jersey, June 1990.

[22] F. Stein and G. Medioni. Structural Indexing: Efficient Three Dimensional Object Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, January 1992. (to appear).

[23] F. Ulupınar and R. Nevatia. Recovering shape from contour for constant cross section generalized cylinders. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 674–676, Maui, Hawaii, June 1991.

[24] A. Witkin and J. Tenenbaum. On the role of structure in vision. In J. Beck, B. Hope, and A. Rosenfeld, editors, *Human and Machine Vision*, pages 481–543. Academic Press, New York, 1983.

Matching Complex Images to Multiple 3D Objects using View Description Networks

J. Brian Burns and Edward M. Riseman

Computer and Information Science Department
University of Massachusetts at Amherst *

Abstract

This paper addresses the problem of effectively matching a single 2D image of a potentially cluttered scene to a library containing multiple polyhedral objects. In our approach to recognition, the search for matches to 3D objects from a multiple object library is optimised by generating descriptions of the projections of the objects from expected views, organising all the descriptions into a single network representation, and then, during the *recognition phase*, finding matches between the resulting *view description network* and the input image. Our design for the recognition phase process is presented and demonstrated on images containing multiple objects and outdoor scenes. The efficiency of the system and the general approach are discussed.

1 Introduction

Our research objective is a system capable of recognising modelled polyhedra from 2D images of cluttered scenes. The system is presented with a library containing multiple objects and an image of a scene that may contain any combination of the objects in arbitrary positions with respect to the viewer. The goal of the system is to find the correct 3D matches to all objects in the scene for which there is sufficient evidence in the image, where a *3D match* is an assignment of model features to image features and the estimated coordinate transformation between model and camera (pose) that best aligns the assigned features. The recognition system must be designed to find the correct 3D matches in an efficient manner. A 3D match is completed and verified by the potentially costly process of searching for image line segments that comprise sufficient evidence for the match. In addition, the number of possible 3D matches can be very large; for the images and objects studied in this paper, there are tens of billions. Therefore, it is essential for the efficiency of the system that the number of 3D matches selected for completion be minimized.

In our approach, the search for matches to 3D objects from a multiple object library is optimized by generating descriptions of the projections of the objects from expected views, organising all the descriptions into a single network representation, and finding matches between the resulting *view description network* and the input image. The contributions of our research have been in the automatic compilation of network descriptions of object *views* [4, 6] and the analysis of the usefulness of 2D features for 3D object discrimination under view variation [5, 6]. In the latter, it was established that there does not exist a feature (function of the projection) that is view-invariant for *arbitrary* point sets. Since view invariants are not guaranteed for any given library of 3D objects, it is imperative that systems be developed that are capable of utilising view-varying features as well [2, 6]. This paper discusses effective strategies for matching view description networks to images and further motivates the overall approach.

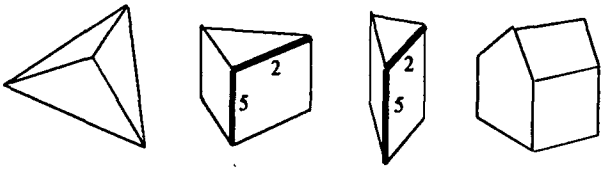


Figure 1: Objects used to demonstrate the system.

2 The description network approach to object recognition

Objects in a library, such as in Fig. 1, will typically have a variety of similarities and differences which must be considered while optimizing the selection of the correct matches. In general, objects can be differentiated using combinations of two types of features: metric and structural. A *metric* feature is any measurement such as length and angle that can be defined for a set of projected elements in the image, which in our task are 2D line segments. Clearly, this type of feature can be useful for discrimination; for example, the two triangular prisms that have different proportions. This differentiating property can be measured in terms of the length ratio of the bold line segments labelled 2 and 5, which is capable of distinguishing the two prisms from almost all of their views [5, 6].

Structural features represent how elements that make up an object, such as straight line segments, are connected or otherwise organized. For example, in Fig. 1, some objects have line segments assembled into triangles

*This research was supported by the Defense Advanced Research Projects Agency (via TACOM) under contract DAAE07-91-C-R035.

and others into pentagons or parallelograms, and these assemblages themselves are connected into larger structures. Clearly, the identification or detection of these structural features can help in the discrimination of the objects. The identification of structural features in the image is important in another way: matched structural descriptions provide a context in which metric features can be measured and meaningfully used for discrimination. For example, the length ratios considered useful in Fig. 1 are meaningless if we do not know *which* pair of image segments are to be measured. There may be an enormous number of ordered pairs of line segments in a cluttered image, and each pair produces its own length ratio. The image segment pair whose ratio provides the discriminating length-to-width proportion must be identified, and this can be done by first matching a structural description.

The importance of structural features for discrimination must be stressed, for their use has a fundamental effect on the design of a multi-object recognition system. Since identifying structural features in the image means searching for matches to non-trivial structural descriptions, an important part of the object discrimination process is an optimized search for these matches. Organizing the structural descriptions is an important step towards optimizing this search, an approach stressed in this study and related work [1, 7].

In our design, object information is organized into *description networks* where parts or geometric aspects shared by objects are explicitly represented as nodes in a network, with direct or indirect links to all the objects characterized by them. Objects are then recognized via the network by a process referred to here as *recursive indexing*. In this strategy, indexing of object information takes place in stages; each indexing step identifies important substructures (parts) in the image which are in turn used as structural features to index more complex descriptions, until descriptions to specific objects are indexed and successfully matched.

This approach to recognition is in contrast to two other important recognition strategies that have recently seen increased development: the single-level, *geometric hashing* methods [13, 14], and the use of *interpretation* trees [9, 10]. Recursive indexing could be a significant improvement over the unstructured, or single-level, hashing methods. In single-level systems, simple image feature combinations are used to directly index the objects in the library. As indicated by the experimental work reported in [14], the object-specificity of the features greatly affects the number of objects retrieved, and the simple combinations of features used in their experiments were not sufficient to avoid a saturation effect (where the number of objects retrieved grows at least linearly with the size of the object library). The specificity problem may become more manageable using a recursive, multilevel procedure, primarily because combinations of structural parts identified in earlier steps could provide the system with a rich set of composite features for indexing at the next step.

It is also important to distinguish the recursive indexing of descriptions in a network from the use of interpretation trees. An interpretation tree search when clutter is present in the image may be prohibitively inefficient [9]. The recursive indexing strategy supported by networks, however, has a fundamentally different behavior that may mean much greater efficiency. Instead of methodically searching a large portion of a tree for possible interpretations, the evidence from matches to multiple parts and the *convergent* structure of the description network are used together to provide a potentially focused search for the correct interpretation. The experiments in cluttered images presented in this paper provide some demonstration of this.

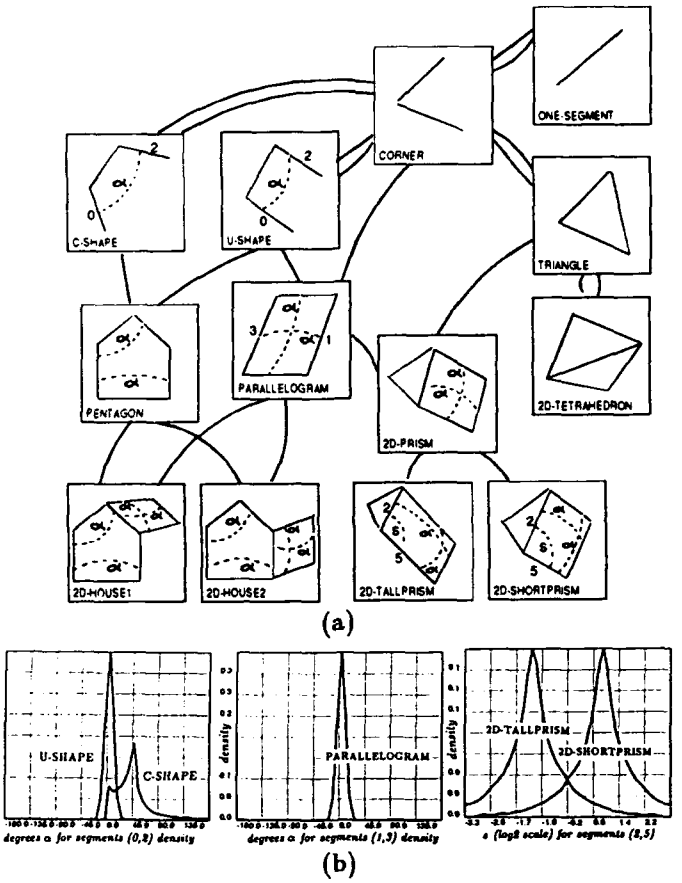


Figure 2: View description network for objects in Fig. 1. (a) Each 2D model node in the network is represented in the figure by a line-segment example that satisfies the 2D model. Dashed lines connecting line segments indicate metric features that are functions of the pair of specified line segments. (b) The feature probability densities given the different 2D models discriminated by them. Each density function is stored in the appropriate 2D model node and is inherited by all of its successors in the network.

3 View description networks

View description is an important approach to recognising 3D objects in 2D images. In this method, prior to recognition, descriptions of the projections of each object from distinct views are generated; then, during recogni-

tion, these 2D expectations are matched to the image. A view description may be valid for the object's projection over a range of views, as long as there is a description associated with each view and the descriptions of each object are distinct from those of other objects. This method is useful since it does not depend on reliable 3D scene reconstruction. It also facilitates the use of image information sufficient for recognition [2, 5] and the efficient matching of *multiple* 3D objects to cluttered 2D images through the organization of the view descriptions into networks. Similar approaches can be found in [7, 8], and also in [10], though the system of Ikeuchi is not for single intensity images.

In our system, a view description is a relational graph that represents the discriminating features for some object and range of view. Each distinct element in the graph represents a line segment, and the relations (arcs) are features defined over the associated line segments. The relation stores the feature type, such as line segment length ratio s or relative orientation α, and the probability densities of the feature given the associated pair of object line segments and a uniform sampling across the range of view.

The view descriptions are organized into a network, where each terminal node corresponds to an object-specific view description. The view descriptions are recursively built up from smaller, simpler relational graphs associated with intermediate nodes in the network via *combination* and *specialization* links. We will refer to the information stored in any node as a *2D model*. Fig. 2 shows a network automatically constructed to recognize the objects in Fig. 1. The combination link specifies how a set of part descriptions are combined into more complex, object-specific ones; currently, combinations are formed of pairs of parts. For example, in Fig. 2(a), the model 2D-PRISM is represented as a combination of 2D models TRIANGLE and PARALLELOGRAM, which are isomorphic to two of its subgraphs. Specialization links between a 2D model node and its network successors specify the addition of new relations to the 2D model; in other words, new features (see Fig. 2a, dashed labelled arcs) and their probability density functions given the object associated with each successor node (see Fig. 2b). For example, the 2D-PRISM model is associated with two objects (*short-prism* and *tall-prism*), and each is assigned a successor containing a new relation: the feature s for element pair $(2, 5)$ and its density function for the relevant object. (New relations can also be added during combination, see Fig. 2b.) Discrimination of the objects based on the stored probability information is discussed below in Section 4.2.

4 Matching images to view description networks

Given a view description network, the goal of the recognition phase of the system is to identify objects in images by efficiently searching for matches between image line segments and object view descriptions (2D model nodes) stored in the network. It was argued above that *recursive indexing* is an effective way to recognize using a description network. This approach is best understood as recursive 2D match extension followed by 3D match completion and verification. *Match extension* is the creation of a match to a 2D model (a *2D match*) from matches to its predecessors in the network, where a 2D match is the assignment of the line elements in the 2D model to line segments detected in the image. For example, given the network in Figs. 2(a) and 3(a), matches to TRIANGLE and PARALLELOGRAM (Fig. 3b) can be combined into a match to 2D-PRISM (Fig. 3c).

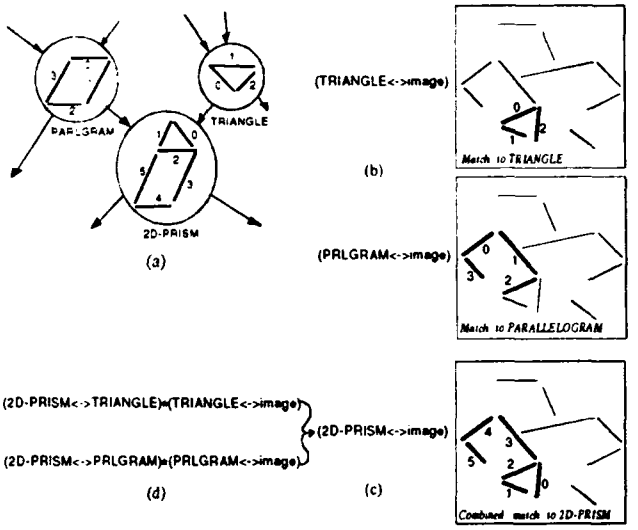


Figure 3: An example of match extension. (a) Relevant portion of Fig. 2. (b) Matches to TRIANGLE and PARALLELOGRAM. (c) A match to their common successor 2D-PRISM, created by (d) composing each predecessor node match with the model-to-model maps specified in the network and then merging the two resulting maps.

The design of the recognition system follows naturally from the recursive nature of matching to view description networks. The process is initialized by detecting line segments in the image and generating promising matches to the initial, simple 2D model nodes in the network. The system then searches for correct matches to the more complex 2D models, and eventually to the 3D models, by iteratively executing the following three steps:

1. *Extend* or *verify* the selected 2D matches, depending on the type of match.
   (a) Matches to 2D model nodes in the terminal portions of the network are associated with 3D objects; when selected, the system attempts to verify them by computing the 3D match.
   (b) Otherwise, the system attempts to extend the match to a more complex 2D model node match.
2. *Evaluate* and *incorporate* the resulting 3D or 2D matches into the current state of the system.
   (a) All new *2D* matches are added to the pool of matches that *may* be extended or verified in future cycles.

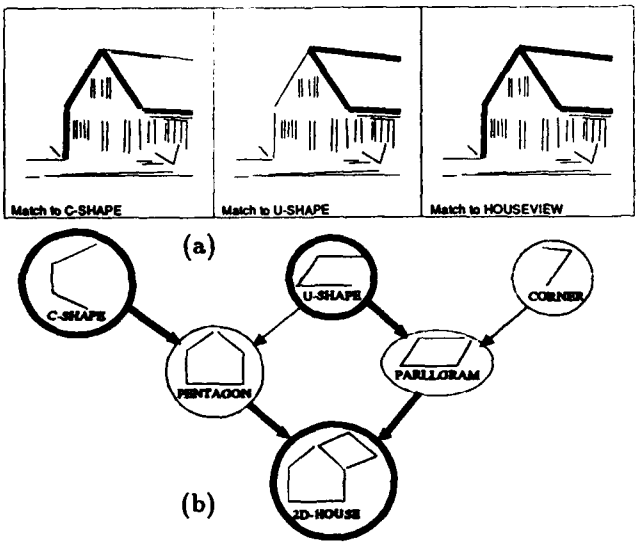


Figure 4: An example of extension to an *indirect* successor in the network of Fig. 2; see text. (a) Matches to C-SHAPE, U-SHAPE and 2D-HOUSE. (b) A portion of the network, with relevant nodes and links in bold.

(b) If a *3D* match is verified, it is output from the system, and competing 2D model matches are eliminated. A match is competing if it assigns the same image segments.

3. *Select* the best 2D matches for extension and verification in the next cycle.

In the experiments, the process was made to terminate on discovery of all the correct 3D matches. The organisation of the system into the above three steps naturally structures the discussion into three parts: 2D match *extension*, *evaluation* and *selection*. Verification by 3D match completion is also clearly important, but is outside the scope of this paper; an implementation of the 3D pose algorithm of Kumar[11, 12] was used in the demonstrations.

4.1 Match extension

As discussed above, the matching of descriptions organised into networks assumes the form of recursive extension. Given a pair of existing 2D matches, their extension has two steps: retrieve 2D models to which they can be extended, and then, compute the line segment assignments for the new, extended match as in Fig. 3.

Complications can occur when key parts of an object's projection, or important relationships between the parts, are poorly represented in the image. The network contains an idealised description of the object projections, with ideal parts and relations between these parts. If the actual representation of a part in the image is poor, then a simple, step-by-step, recursive extension could be difficult and the extension process must be made more adaptable. The matching of a house image in Fig. 4(a) provides an example of this. (See Fig. 7 for the digital image.) Given the network in Figs. 2(a) and 4(b), a 2D-HOUSE model is made up of two parts, PENTAGON and PARALLELOGRAM, which are in turn made up of the simpler models, U-SHAPE, C-SHAPE and CORNER. The extension of the U-SHAPE match in Fig. 4(a) to a match of its *direct* successor, PARALLELOGRAM, via combination with CORNER is not possible since the image frame excludes the part of the projection associated with the desired CORNER match. Without the evidence associated with this CORNER match, there is no way to know whether the U-SHAPE match should be interpreted as part of a PARALLELOGRAM or PENTAGON match, both being successors of U-SHAPE (see Fig. 4b). However, a viable, unambiguous extension to the *indirect* successor 2D-HOUSE is possible by combining the U-SHAPE match with the available C-SHAPE match shown in Fig. 4(a). The importance of indirect predecessor extension for matching can be appreciated in the application of the recognition system to cluttered and corrupted images as demonstrated later in this paper. To support indirect predecessor extension, our system is designed to index 2D models given pairs of matches to their fragments, that is, their indirect predecessors. During the compilation phase of the system, 2D models and potentially useful pairs of their indirect predecessors are hashed into an *extension table* for rapid indexing during the recognition phase. In addition, to facilitate the extension operation, the compilation phase process also pre-computes and stores the model line segment mappings between the 2D models and useful indirect predecessors.

4.2 The evaluation of matches to 2D model nodes

Once a 2D match has been generated, its priority for extension or 3D verification needs to be determined for effective control. An important factor in assessing this priority is the probability that the match is correct, which is estimated in two steps. First, all of the metric and structural features represented in the 2D model are measured; then, the values of the measured features and the conditional density functions stored in the models are used to estimate the posterior probability of the match.

The metric and structural features used in the match evaluation are measured with respect to an approximate reconstruction of the object's projection consistent with the match. Fig. 5 shows an example of this. The structural features in the 2D model specify how the matched line segments should be connected together. The reconstruction enforces the specified segment connectivity, while minimizing the error between the detected fragments and reconstructed lines [6]. Since the errors reflect the degree of actual connectivity of the detected line segments, they are used as measurements (indications) of the structural features for match evaluation. The *metric* features, such as line segment length ratio s or relative orientation α, are measured with respect to the reconstructed lines.

A match is an assignment of 2D model lines to segments in the image, and this assignment is one possible interpretation of the image segments among many. These alternative interpretations are classes, and the

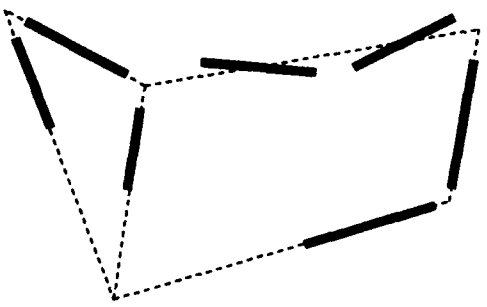


Figure 5: An example of line segment reconstruction during 2D model matching. Bold lines: line segments detected in an image of a tall prism. Dashed lines: reconstruction of the projection given a 2D-PRISM match.

problem of estimating the probability that a given match is correct can be treated as a problem in Bayesian classification:

$$P(\omega_1 \mid \vec{f}) = \frac{p(\vec{f} \mid \omega_1)P(\omega_1)}{\sum_{j=1}^{n} p(\vec{f} \mid \omega_j)P(\omega_j)}$$

where ω_1 is the class associated with the 2D model match being evaluated, the $\omega_j, j > 1$ are alternative interpretation classes, and $\vec{f}$ is the vector of measured features. This formulation is useful and straight-forward; however, it requires the determination of the alternative interpretation classes $\omega_j, j > 1$, and, for all classes $\omega_j, j \geq 1$, the class priors $P(\omega_j)$ and the conditional density functions $p(\vec{f} \mid \omega_j)$.

For a given match, there can be a very large number of alternative interpretations of the same set of image line segments; however, these interpretations can be usefully organized into a small number of classes. For each alternative class, the actual match being evaluated is considered *incorrect*, and in our formulation, each alternative interpretation class is associated with a different type of matching mistake that could have produced the incorrect match. The match being evaluated is generated by extending a pair of other matches, say M_1 and M_2, and this implies the following classes of alternative interpretations (matching mistakes): (1) M_1 and M_2 are both correct but the given extension is not (i.e., some *other* extension is); (2) M_1 and M_2 are correct but they cannot be combined into *any* valid extension; (3) M_1 is itself incorrect; (4) M_2 is incorrect; and (5) *both* M_1 and M_2 are in correct. These distinct classes have different priors and class conditional densities for the features.

The priors assigned to each class are derived from view analysis and estimates of match error rates. Assuming a uniform distribution for views, the prior probabilities for the given 2D match and classes of type (1) are a function of the visibility of the different 2D models; i.e., the fraction of views for which they are valid. The alternative interpretation classes associated with the other matching mistakes (2-5) are assigned priors that reflect the expected rate at which these mistakes occur. These error rates have not been rigorously estimated; however, this does not seem to have caused serious problems in the experiments.

Estimating the probability that a 2D match is correct also requires the conditional density functions for the features given each of the interpretation classes defined above. For all the classes considered, the features are assumed to be independent. As discussed in Section 3, the density functions for the *metric* features given a correct match to the 2D model are represented with the model (see Fig. 2b). The *structural* features are measured in terms of the reconstruction fit errors. Given that the 2D model match is correct, the error in the fit is strictly a function of the inaccuracies of the detected image segments, and the associated *detection error* density function is assumed to be a Gaussian with sero mean. When the match is *incorrect*, structural features (fit errors) are largely due to the incorrect model line assignments, and the associated *assignment error* density function is a much wider Gaussian. For some of the alternative interpretation classes defined above, not all of the match is considered incorrect, and thus not all of the associated reconstruction fit errors are modelled as assignment errors. For example, in class (3), the part of the match extended from M_1 is considered incorrect but not the part from M_2. The probability densities of the fit errors for these two different parts of the match thus reflect assignment and detection errors respectively. A more complete treatment can be found in [6].

4.3 Control: Selecting 2D matches for extension and verification

As discussed above, the overall form of the matching system is that of a search for correct 3D matches by iterative 2D match extension and verification. Therefore, the system's behavior is controlled by the selection of matches to extend or verify during each iteration. While verification is the transformation of a *single* 2D match into its 3D counterpart, match extension involves the combination of *pairs* of matches. It is computationally prohibitive to select them by evaluating and assigning a priority to every pair of existing matches. Thus, for each iteration, the selection proceeds in two steps: (1) select a set of *individual* matches, each with a high likelihood of leading to the correct interpretation, and (2) for each selected match that is to be extended, retrieve *partner* matches for combination.

Given the above process, it is clear that not all of the possible combinations of a match are attempted when it is selected for extension. Even if the combinations tried seem the most promising, the correct one may have been excluded. It is thus desirable to be able to *re-select* a match in another iteration, and retrieve a new set of partners to combine with it.

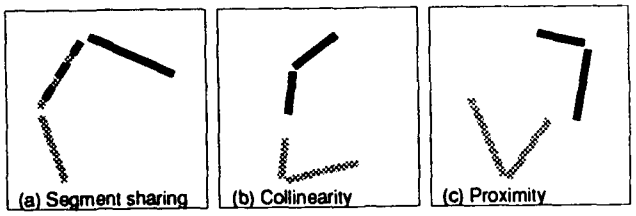


Figure 6: Different pairs of CORNER matches demonstrating three perceptual organization factors important for assessing the potential usefulness of a match pair as a combination. For each example, line segments associated with one match are shown in black, those with the other, in gray, and those with both, in alternating black/gray. The factors exemplified are (a) connectedness (segment sharing), (b) good continuity (collinearity), and (c) proximity.

4.3.1 Selecting individual matches to extend or verify.

The set of individual matches selected during each iteration should satisfy some combination of the following two criteria: the matches must be promising candidates for extension or verification, and the selected set must be distributed about the image in a way that is advantageous for recursive matching to view description networks.

The first factor, selecting candidates for extension or verification with high likelihoods of leading to the correct 3D interpretation, is clearly a function of the probability that the 2D match itself is correct. However, for those matches selected for extension, this likelihood is also a function of whether or not the match can be successfully combined with another match. From the above discussion, it is clear that a match may have already been selected in an earlier iteration, and thus some of the most promising combinations with it will have been attempted already. It seems reasonable to assume that the chance that the correct combination has *yet* to be generated goes down each time a match is re-selected for extension. The priority assigned to a match is a product of the probability that it is correct and the probability that it can yet be successfully combined.

The second factor, matching an image to an object represented by a view description network, is best satisfied if the match extension activity is distributed over the projection of the scene object. In this way, matches to different parts of the projection will have a greater chance of being available for important combinations at roughly the same time. The following is the method used for distributing the selected matches: (a) initialize the matching system by generating primitive 2D model node matches in different portions of the image; and (b) for subsequent iterations, select the highest priority match in the neighborhood of each of the matches last extended. In our system, the *neighborhood* of a match includes itself and all matches related to it through extension.

4.3.2 Selecting match combination partners

Once a match is selected for extension, the system searches for other matches that provide promising combinations. Each time a match is re-selected for extension, a new set is sought for combination, in order of most promising sets first. A pair of matches is a promising combination if the resulting extension to a new 2D match has a high probability of being correct. In a cluttered image, it is easy to select a pair of matches whose individual probabilities are high, but their *combination* has a low probability of being correct; thus, it is important to be able to rank candidate pairs based on how they combine. An important set of heuristics for ranking combinations in order of their probability of being correct can be found in psychological studies of perceptual organization. Generally, image features exhibit compelling perceptual organization if they appear to the viewer as parts of the same object.

The three perceptual organization factors exemplified in Fig. 6, connectedness, good continuity, and proximity, are important for assessing match combinations and have been incorporated into our recognition system. *Connectedness* is exhibited when two matches share the same image line segment. If these two matches are correct, there is a high probability that they are of the same object's projection. Two matches exhibit *good continuity* if a pair of image line segments, one from each match, are close and approximately collinear. Given good continuity, the visual organization is very compelling, though not with the strength of connectedness. Finally, two matches are considered proximate if their associated image line segments are close. Proximate matches do appear more likely to be parts of the same object than more distant pairs, but this factor is clearly not as compelling as the others.

The combinations of a given match are generated in order of the compellingness of the grouping heuristic involved: all combinations exhibiting connectedness are of highest priority, then those with good continuity, and for subsequent passes, the rest are ordered by proximity. Given this scheme, the extensions of a match are attempted in roughly best-first order. In addition, the retrieval of the desired match combinations, given each of the factors, can be made efficient by suitable image and match data base organization, as has been implemented in our system.

5 Matching experiments

Our matching system has been applied to the recognition of objects in real, digital images. For each image, the same initialization procedure was followed. First, lines were detected [3] and filtered by length (> 10 pixels) and intensity contrast (> 5 gray levels). Next, all matches to CORNER with low gap error ($< 12.5\%$ of line length) and low overshoot of the intersection ($< 3.5\%$) were found. Finally, redundant CORNER matches due to redundantly represented object edges were removed by the following filter: if a line segment is matched to CORNER multiple times, and the other segments in these multiple CORNER matches are next to and parallel with each other, then select only the best one.

The first three images in Fig. 7 are of scenes containing multiple objects. These were matched to a network

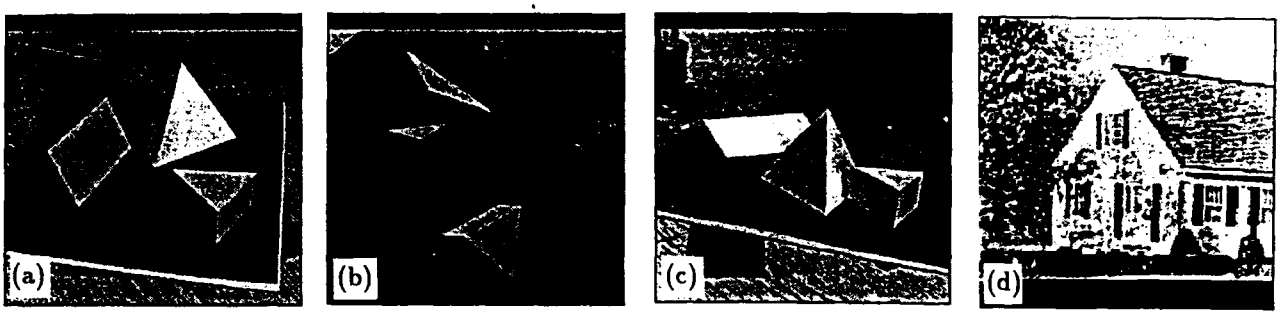


Fig. 7 Images of scenes containing multiple objects and an outdoor scene. Images: (a) *separated-objects*, (b) *top-scene*, (c) *side-scene* and (d) *outdoor-scene*

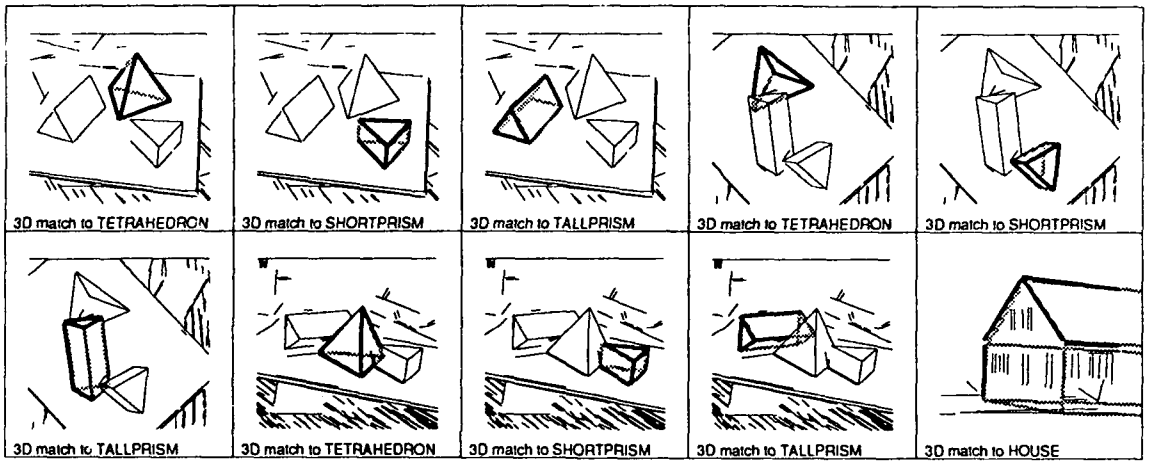


Fig. 8 The resulting correct 3D matches for the images. The line segment assignments of each match was generated by the network matching process and are indicated by the thick black lines. The thick gray lines are projections of the object from the 3D pose estimated by the algorithm of Kumar, given the segment assignments shown.

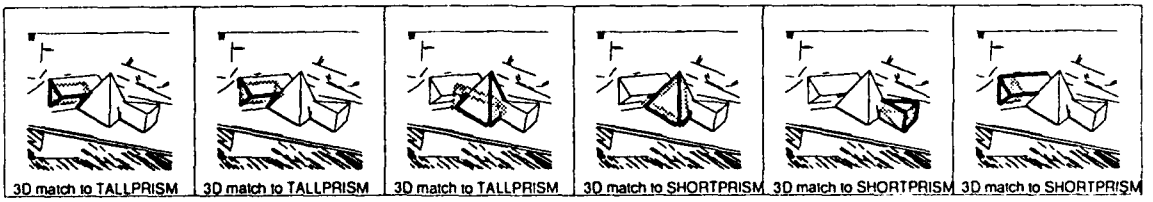


Fig. 9 The incorrect 3D matches generated by the system and rejected by the verifier. All of the incorrect 3D matches are from the image *side-scene*; the matches are represented as in the previous figure.

Statistics	Images				
	Separate-objects	Top-scene	Side-scene	Outdoor-scene	Average
object library size	3	3	3	4	3.25
image segments	63	77	142	41	80.8
3D matches possible	2.5×10^9	7.1×10^9	1.6×10^{11}	3.6×10^8	4.3×10^{10}
3D matches generated	3	3	9	1	4.0
3D matches per correct match	1	1	3	1	1.6
number of match iterations	4	4	5	4	4.25
ave height of network	4.7	4.7	4.7	4.8	4.73
iterations per network level	.86	.86	1.07	.83	.91
3D and 2D matches generated	230	264	379	137	252.5
ave # line segs per match	2.6	2.5	2.3	1.9	2.37

Table 1 Statistics for matching experiments. Each column reports the statistics for a different image, except for the last column which is the average over all runs. The statistics are explained in the discussion section.

for the three objects in the scenes; it was identical to the one shown in Fig. 2, without the house view descriptions. The system was also applied to the last image in Fig. 7 of an outdoor house scene. The system searched for matches between this image and all four objects shown in Fig. 7, using the view description network shown in Fig. 2. All of the correct 3D matches were found for each image; Fig. 8 shows the original hypothesized 3D matches as thick black lines and the resulting projection given the estimated 3D pose in thick gray lines. The system also hypothesized and attempted to verify some incorrect 3D matches, shown in Fig. 9. Note that there were no incorrect 3D matches generated for the images *separated-objects*, *top-scene* and *outdoor-scene*, and only six for *side-scene*. The first four of the incorrect matches were due to false line-segment junctions and accidental parallels in projection (or shadows), which produced erroneous small 2D matches of high probability. In spite of such complications in the image, the matching system as a whole seems effective.

Table 1 shows some useful statistics for each matching trial and the average across all of them (last column). For the images and objects studied here, a confident 3D match typically requires the assignment of five model line segments, making the possible 3D matches per image number in the tens of billions. In spite of this, the number of 3D matches actually hypothesized is very small, averaging 1.6 per correct 3D match. Another illuminating statistic is the number of iterations that the system runs before finding all of the correct matches. Including the initialization and final verification steps, the average number of iterations is 4.25. In relation to the average height of the view description network, this is a good result. The average height of the network roughly represents the number of steps in a network-directed construction of the 3D match, starting from an unmatched set of image line segments. For the network used in this study, the average height is 4.7, which is higher than the actual average number of iterations used by the system to generate the correct 3D matches. In part, this reflects the fact that the system sometimes performed match extensions to *indirect* successors in the network and thus avoided some of the construction steps. It also reflects the utility of the match combination approach in general. As argued in Section 2, the evidence from matches to multiple parts and the convergent structure of the description network are used together to provide a potentially focused search for the correct interpretation. The results presented here help demonstrate this potential.

The total number of 3D and 2D matches generated also seems reasonable. This is especially true when one considers the number of line segments in the image and the average size of the matches (number of assigned model lines). In the trials presented here, matches to the two simplest 2D models, LINE-SEGMENT and CORNER, make up the majority of the total, and another large portion of the total is made up of matches to 2D models that are almost as simple: the three-segment matches to U-SHAPE, C-SHAPE and TRIANGLE. For the images tested, the system consistently converges on the correct interpretation without generating many 2D or 3D matches of size greater than three segments – even though there can be many initial matches to the smaller 2D models.

6 Summary and conclusions

In our approach to recognition, the search for matches to a multiple 3D object library is optimized by matching *view description networks* that are automatically compiled from the library. The contribution of the research described in this paper is the development of effective strategies for the network matching process. The experiments show that a recognition system based on view description networks is capable finding the correct matches to 3D objects in complex images with a potentially high level of efficiency.

References

[1] Biederman, I. (1985) "Human Image Understanding: Recent Research and a Theory", CVGIP 32, p29-73

[2] Ben-Aire J. (1990) "The Probabilistic Peaking Effect of Viewed Angles and Distances with Application to 3D Object Recognition", IEEE PAMI 12.8, p760-774

[3] Boldt, M., R. Weiss and E. Riseman (1989) "Token-based Extraction of Straight Lines", IEEE SMC v.19.6. pp. 1581-1594.

[4] Burns, J.B. and L.J. Kitchen (1987) "Recognition in 2D images of 3D Objects from Large Model Bases using Prediction Hierarchies", Proc. IJCAI-10.

[5] Burns, J.B., R. Weiss and E. Riseman (1990) "View-variation of point-set and line-segment features", DARPA IUW, Also to appear in IEEE PAMI (1992)

[6] Burns, J.B. (1991) "Matching 2D Images to Multiple 3D Objects Using View Description Networks", PhD. Thesis, UMass Amherst.

[7] Draper, B., J. Brolio, B. Collins, A. Hanson, and E. Riseman (1989) "The Schema System", IJCV, v.2.3, p.209-50.

[8] Gigus, Z. (1990) "Computing the Aspect Graph for Line Drawings of Polyhedral Objects" IEEE PAMI, v.12.2, pp. 113-122.

[9] Grimson, W.E.L. (1991) *Object Recognition by Computer: The Role of Geometric Constraints*, MIT Press, Cambridge, MA.

[10] Ikeuchi, K. (1987) "Precompiling a Geometrical Model into an Interpretation Tree for Object Recognition in Bin-picking Tasks", DARPA IUW, pp. 321-339.

[11] Kumar, R. and A. Hanson (1989) "Robust Estimation of Camera Location and Orientation from Noisy Data Having Outliers", IEEE Work. on Interp. of 3D Scenes, pp. 52-60

[12] Kumar, R. and A. Hanson (1990) "Sensitivity of the Pose Refinement Problem to Accurate Estimation of Camera Parameters", IEEE ICCV, pp. 365-369.

[13] Lamdan, Y. and H.J. Wolfson (1988) "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme", Proc. ICCV, pp. 238-249.

[14] Stein, F. and G. Medioni (1990) "Efficient 2D Object Recognition", Proc. ICPR, pp. 13-17.

Noise Resistant Projective and Affine Invariants

Isaac Weiss*
Center for Automation Research
University of Maryland
College Park, MD 20742

Abstract

Projective invariants are shape descriptors which are independent of the point of view from which the shape is seen, and therefore they are of major importance in object recognition. They make it possible to match an image of an object to one stored in a data base without the need to search for the correct viewpoint. In this paper we obtain an invariant representation ("signature") of a general curve. The calculation is local and does not suffer from the occlusion problem of global descriptors. The affine transformation subset is also treated. To make the method robust, we have developed differentiation techniques which give much more reliable results than previous ones. These differentiation methods are useful in many other applications as well.

1 Introduction

A major problem in object recognition is the fact that the same shape can be seen from different points of view, resulting in different images. In order to compare a given image to one that is stored in a library of images, existing methods had to search in a multidimensional parameter space to find the appropriate "pose", or point of view. Invariants of shapes, being independent of the point of view, free us from this search and allow direct matching between the observed and the stored images.

Projective invariants were a very active mathematical subject in the latter half of the 19th century. However, in vision only one projective invariant, the cross ratio of four points on a line [7], was used until recently.

Projective invariants of curves and surfaces were first introduced in vision by this author [13]. The two main kinds of invariants, the algebraic and differential invariants, were reviewed in that paper, which pointed out their usefulness for object recognition. Algebraic invariants were then successfully applied to industrial objects in [8]. Recognition of occluded surfaces using invariants was treated in [3], and semi-differential invariants were used in [12] and [1]. Other papers are listed in the references.

*This work was supported in part by ONR Grant N00014-91-J-1222

Algebraic invariants are well suited to use with algebraic shapes, i.e. shapes that can be expressed as a 2-D polynomial $f(x,y) = 0$, e.g. conics. The polynomial coefficients are obtained by fitting the appropriate polynomial to the whole visible shape and calculating the invariants from the polynomial coefficients. These invariants have several problems: first, most shapes are not algebraic, making it hard to fit simple polynomials to them. Second, the algebraic method is global, requiring dealing with whole shapes. Like any global descriptors, they are vulnerable to occlusion problems.

Differential invariants overcome these problems because they are local, i.e. invariants are found for each point on the curve. Furthermore, the curve can be quite general.

In the Euclidean case it is common to plot the curvature against the arclength, both Euclidean invariants, to obtain a "signature", or an invariant curve. Given this signature, we can reconstruct the original curve up to a Euclidean transformation. Similar signatures can be obtained for both affine and projective transformations, which are our interest here. This is because of a general *completeness* property.

The completeness property of differential invariants can be described as follows. Given a plane curve and a transformation group, there are two independent invariants of the transformations at each point of the curve. These invariant functions contain all the information about the curve, except for the transformation to which they are invariant. Accordingly, given two invariants for each curve point, we can determine the original curve up to a transformation belonging to the group.

More accurately, the following theorem holds [9, p. 144]:

Theorem. *All differential invariants of a (transitive) transformation in the plane are functions of two invariants of the lowest order and their derivatives.*

Thus, given a curve, one can find a corresponding invariant curve, its signature, that describes it uniquely, except for the relevant transformation.

In this paper we present a method of object recognition based on the above property. The transformations we are interested in are the affine and projective ones, because they are induced by changing the point of view. At each point of the given curve we calculate two invariants, I_1, I_2. We plot these numbers as a point in

an "invariant plane" whose coordinates represent invariants. In effect we plot one invariant against the other. In this way the given curve maps into an invariant signature curve in the invariant plane. The signature uniquely identifies the curve regardless of the point of view.

It is useful to know the amount of information that needs to be obtained from the image. To find invariants, we have to eliminate the information in the image which is specific to the coordinate system. For example, given a pencil that can move or rotate on a table, the position of the pencil and its orientation are not invariant but its length is a Euclidean invariant. Given the coordinates, say of the ends of the pencil, we can eliminate the position and orientation and calculate the distance. Thus from the four measured coordinates we have eliminated the three Euclidean transformation parameters and found one invariant.

Similar arguments apply for other transformations. In the projective case, we want to eliminate eight parameters of the transformation, so the number of coefficients to be obtained from the image should exceed eight. It is easy to apply this to algebraic shapes. For example, one conic does not contain enough coefficients but two conics do, with ten independent ones, and so do a cubic (nine coefficients) and a quartic (14).

In the differential case the same arguments apply. However, in traditional methods of obtaining invariants [16], a complication arises because the curve is represented parametrically as $x(t), y(t)$, with an arbitrary curve parameter t. Wilczynski's method requires the eighth derivative (of both $x(t)$, $y(t)$) to find two invariants to both the projection and the change of parametrization, for a total of 18 quantities. If we disregard the parameter problem, we need only the fourth derivatives of x, y, namely only ten numbers. Thus, if we could overcome the parameter problem, the number of data quantities needed will be no more than required by an algebraic method.

The parameter problem can be avoided from the outset because the parameter is not part of the geometry of the curve; the coordinates x, y of each point are sufficient to determine the curve. The parameter is an arbitrary function introduced for convenience, so it is clear that one can improve on these methods. In this paper we present an approach which avoids the parametrization problem and enables us to find two local invariants with only the ten quantities that are needed from a purely geometrical point of view. In this way we combine the advantage of the differential method, its locality, with that of the algebraic method, which does not need a curve parameter.

Of course the fourth derivative, or similar quantities needed in our new invariant method, are still high by computer vision standards, and common methods completely fail to obtain them. We have found a smoothing and differentiation method that gives good results for high derivatives. In our experiments the error in the derivatives and in the invariants was no more than the error in the data. Obviously the method has applications beyond invariants.

In the following sections we review the previous methods for deriving projective and affine differential invariants, then we describe our new methods for finding invariants and for obtaining derivatives, and finally experiments are presented.

2 Wilczynski's method

In this section we describe the method developed by Wilczynski [1906] who obtained closed form formulas for a complete set of differential projective invariants of curves.

A projective transformation in the plane can be written in Cartesian coordinates as

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{pmatrix} = \frac{1}{xT_{31} + yT_{32} + T_{33}} T \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

where T is a constant matrix. The factor multiplying T in this equation contains the coordinates, so the transformation in non-linear and also leads to infinities when it vanishes. To avoid dealing with these problems it is common to generalize the treatment by working in homogeneous coordinates $\mathbf{x} = (x_1, x_2, x_3)^t$ and write the transformation as

$$\tilde{\mathbf{x}} = \lambda(\mathbf{x}) T \mathbf{x}$$

with $\lambda(\mathbf{x})$ being an arbitrary factor, that can be different at each point $\mathbf{x}$. This can be generalized to curves in n-dimensional homogeneous space.

A curve in n-D can be written parametrically as $\mathbf{x}(t)$. We want to find quantities at each point of the curve which are independent of both the coordinate system and the parameter t. To find invariants, one can proceed in stages: (i) find invariants to the linear part T of the transformation above, (ii) from these derive invariants to multiplication by λ, (iii) from the latter construct invariants to the curve parameter t. We will proceed to describe each stage.

The basic method of obtaining invariants in this approach is by using derivatives. The advantage of differentiation is that it eliminates constants, which are often associated with the coordinate system. For instance, a straight line can be written as $y = ax + b$, with the constants a, b both depending on the coordinates. However, the equation $y'' = 0$ represents straight lines (and only them) invariantly of any coordinate system. The more general case is more involved but the same principle applies.

Invariance to T can be obtained by taking the derivatives of the curve $\mathbf{x}(t)^{(n)}$ and forming the set of linear algebraic equations

$$\mathbf{x}^{(n)} + \binom{n}{1} p_1 \mathbf{x}^{(n-1)} + \binom{n}{2} p_2 \mathbf{x}^{(n-2)} + \ldots + p_n \mathbf{x} = 0 \quad (1)$$

for the unknowns $p_1 \ldots p_n$, at each point t. For a plane curve ($n = 3$) we have

$$\mathbf{x}''' + 3p_1 \mathbf{x}'' + 3p_2 \mathbf{x}' + p_3 \mathbf{x} = 0$$

with the three unknowns p_1, p_2, p_3. It is easy to see that multiplying $\mathbf{x}$ by a constant matrix T has no influence on the above equation since the matrix factors out. Thus the p_n are invariant to the linear transformation T.

Next, we have to deal with the transformation of multiplying by λ, namely $\tilde{\mathbf{x}} = \lambda(\mathbf{x})\mathbf{x}$. We find quantities P_i, functions of p_i, that are invariant to this transformation. Since P_i are not invariant to transforming the parameter t, they are called *semi-invariants*. For a planar curve we have [16, p. 58]

$$P_2 = p_2 - p_1^2 - p_1' \tag{2}$$

$$P_3 = p_3 - 3p_1p_2 + 2p_1^3 - p_1''$$

The final stage is finding invariants to the parameter t. The transformation in question is $t \rightarrow \tilde{t}(t)$, to which we can find *relative invariants of weight w*, namely quantities that transform as

$$\tilde{\Theta}_w = \frac{1}{(\tilde{t}')^w}\Theta_w$$

These invariants are [p. 59]

$$\Theta_3 = P_3 - \frac{3}{2}P_2'$$

$$\Theta_8 = 6\Theta_3\Theta_3'' - 7(\Theta_3')^2 - 27P_2\Theta_3^2$$

They are relative invariants of weights 3 and 8, respectively.

The question arises of how many independent invariants exist for a curve and to what extent one can reconstruct the curve given the invariants. We have the completeness theorem:

Theorem. *The invariants Θ_3, Θ_8 completely determine a plane curve up to a projective transformation.*

This is a special case of the more general property mentioned in the introduction. From these, other invariants can easily be derived. In particular

$$\Theta_{12} = 3\Theta_3\Theta_8' - 8\Theta_8\Theta_3'$$

Although the invariant set Θ_3, Θ_8 is complete, it cannot be used as is to identify the curve because they are relative invariants, i.e. the transformation contains the unknown $(\tilde{t}(t)')^{-w}$. But we can derive from them absolute invariants ($w = 0$). We can choose

$$I_1 = \frac{\Theta_3^8}{\Theta_8^3}, \qquad I_2 = \frac{\Theta_3^4}{\Theta_{12}}$$

We can now define an "invariant plane", with coordinates I_1, I_2. For each point on the given curve we can calculate the two invariants and draw a point I_1, I_2 in the invariant plane. In this way we obtain an invariant curve, or signature, of the original curve.

Calculating the above invariants requires the eighth derivative with respect to the parameter, and this poses a hard problem from a practical point of view. The invariants Θ_3, Θ_8 were implemented numerically, using simple finite difference methods, in [2]. It was concluded that in this simple implementation the above invariants are quite unreliable and hard to use in practice.

The method of invariants can be made reliable in several ways, of which two are treated here:

i) Calculate derivatives using more sophisticated numerical analysis methods. We have succeeded in reliably obtaining a fourth derivative from numerical data.

ii) Develop methods that need fewer quantities such as derivatives that need to be obtained from the image. We have developed a method that does not require a parametrization of the curve and thus obviates the need for the high derivatives needed in the above invariants.

3 Our New Invariants

In this section we derive semi-invariants requiring fewer derivatives than Wilczynski's. We later derive an entirely different method based on a canonical coordinate system.

Modified Semi-invariants

The semi-invariants written above can be modified to reduce the number of derivatives needed from five to four. From eq. (2) we see that the semi-invariant P_2 contains only the first derivative p_1'. Since the p_i themselves depend on the third derivatives of the curve, P_2 depends on the fourth derivatives. However, P_3 depends on the fifth derivative of the curve. This can be eliminated by subtracting P_2':

$$P_3^\star = P_3 - P_2' = p_2' - 2p_1p_1' + 3p_1p_2 - 2p_1^3$$

Our curve can now be described by P_2, $P_3^\star$ which are invariant to the projection and to change of the arbitrary multiplying factor $\lambda(t)$, and only involve four derivatives. They are not invariant to the change of parameter t.

Canonical Invariants

We now develop a new way to obtain invariants, which is different from Wilczynski's and offers more insight into the geometrical nature of the invariants. So far only semi-invariants are obtained by this method as they depend on the parameter. In the next section we derive full invariants by a different method that does not involve a parameter.

The basic idea is to use a projective transformation in order to go over from a given coordinate system to a "canonical" one, in which some of the derivatives are predetermined. Other derivatives will then become invariants. The concept can be illustrated by examples of simpler transformations. If a 1-D function $x(t)$ is subject to scale transformation in x, we can always fix the scale along x by defining a canonical coordinate $\bar{x}$ such that $\bar{x}'(0) = 1$. The transformation to the canonical system is a simple normalization $\bar{x} = x/x'(0)$. By fixing the scale, other quantities that depend on it such as the second derivative $\bar{x}''(0)$ are also fixed and are now scale invariant. In another example, given a curve in the Euclidean plane, we can move the coordinate system so that the x axis is tangent to a certain point on the curve, $\bar{y}' = 0$. The second derivative $\bar{y}''$ in this system is now fixed (as curvature) and is invariant since we can obtain the same canonical system regardless of which system we started with. We see that by determining some of the derivatives of the system, the others are also determined and become invariant. We generalize this process for the projective case.

We define our canonical system as follows, with non-homogeneous coordinates:

Table 1. Canonical Coordinate System

	f	f'	f''	f'''	f''''
$\bar{x}$	0	1	0	1	0
$\bar{y}$	0	0	1	I_1	I_2

where $\bar{x}, \bar{y}$ are the canonical coordinates, f in the first line is either $\bar{x}$ or $\bar{y}$ and the primes denote derivatives with respect to t. The I_1, I_2 mean that there are no conditions on these derivatives, $\bar{y}''', \bar{y}''''$; they are our desired invariants, as will be shown below. There are eight conditions in all, and they can be satisfied by employing the eight parameters of a projective transformation to arrive at this canonical coordinate system. The following theorem holds:

Theorem. *Given a curve in the canonical system of Table 1, its parameterization around the origin is determined uniquely up to and including the fourth derivative. Conversely, given a curve and its first four derivatives with respect to some parameter t, one can always find a coordinate system, unique for each point, satisfying these canonical conditions.*

The proof of the converse is constructive, and gives the appropriate transformation from the given coordinate system to the canonical one. The details are given elsewhere [14].

These invariants have been calculated explicitly and tested numerically. It was confirmed that they are invariant to the projectivity, but not to a change of the parameter. Full invariance requires either higher derivatives, as in Wilczynski's method, or getting rid of the parameter altogether, as described later.

4 Affine Differential Invariants

Differential invariants can be obtained using Cartan's method of moving frames. Unlike other methods such as Lie group prolongations, it does not require solving differential equations. We will briefly describe the method and apply it to the affine case, following [9].

The basic idea is to find an invariant of a "moving frame". This is a local coordinate system that moves along the curve by the action of a transformation. We will first illustrate it in the Euclidean case.

Given a curve parametrized by its arclength s, we can define a local orthogonal frame at each point, made up of the unit tangent $\mathbf{t}$ and the normal $\mathbf{n}$. This is our Euclidean moving frame. We also have a fixed global frame, with unit vectors $\mathbf{e}_1, \mathbf{e}_2$ along the coordinate axes. At each point s the local frame can be obtained from the fixed one by a rotation matrix $A(s)$ (translation is ignored):

$$\{\mathbf{t}, \mathbf{n}\} = A(s)\{\mathbf{e}_1, \mathbf{e}_2\}$$

The variation of the local frame along s can be written as

$$\frac{d}{ds}\{\mathbf{t}, \mathbf{n}\} = A'(s)\{\mathbf{e}_1, \mathbf{e}_2\} = A'A^{-1}\{\mathbf{t}, \mathbf{n}\}$$

We define the *Cartan matrix* $C(A)$

$$C(A) = A'A^{-1} \tag{3}$$

which is a general definition for any frame transformation A. We now have

$$\{\mathbf{t}, \mathbf{n}\}' = C(A)\{\mathbf{t}, \mathbf{n}\} \tag{4}$$

When we transform the curve both sides of this equation are multiplied by a constant rotation matrix B. This leaves the equation unchanged and thus $C(A)$ is invariant.

In our case A is a simple rotation and it is easy to calculate its Cartan matrix:

$$C(A) = \begin{pmatrix} 0 & \kappa(s) \\ -\kappa(s) & 0 \end{pmatrix}$$

with $\kappa(s)$ being the curvature at s, a well known Euclidean invariant. Eq. (4) can be written explicitly as

$$\begin{aligned} \frac{d\mathbf{t}}{ds} &= \kappa(s)\mathbf{n} \\ \frac{d\mathbf{n}}{ds} &= -\kappa(s)\mathbf{t} \end{aligned}$$

which are the well known *Frenet equations* of plane differential geometry.

Cartan's method generalizes the construction and the invariance of the Cartan matrix $C(A)$. We can represent a curve by a moving frame generated by the matrix A

$$\{\mathbf{a}_1, \mathbf{a}_2\} = A(t)\{\mathbf{e}_1, \mathbf{e}_2\}$$

with $\mathbf{a}_1, \mathbf{a}_2$ being local vectors making up the moving frame. Transforming the curve is equivalent to multiplying all the local frames by the same transformation matrix B, i.e. the frame matrix in the new coordinates system is $\tilde{A}(t) = A(t)B$. We now have the theorem:

Theorem. *Let the (differentiable) frame matrix $A(t)$ belong to a Lie group G. Let B be a constant transformation matrix also belonging to G, which does not change the parameter t. Then $C(A) = C(AB)$.*

In other words, the Cartan matrix $C(A)$ is invariant under the transformation B.

Proof: From the definition (3) of the Cartan matrix we have immediately

$$C(AB) = C(A) + AC(B)A^{-1} = C(A)$$

The last equality is due to the fact that B is constant so its Cartan matrix vanishes.

This theorem assumes that the parameter t is not changed by the constant transformation B. Thus, for the method to work, we have to find an invariant parameter. This is easy to do for a unimodular affine transformation (i.e. one with a unit determinant). In this case we define the moving frame as the two local vectors $c\mathbf{x}_t, c\mathbf{x}_{tt}$ with the normalizing factor $c = |\mathbf{x}_t, \mathbf{x}_{tt}|^{-1/2}$. Thus the frame transformation matrix is

$$A(t) = c\{\mathbf{x}_t, \mathbf{x}_{tt}\}$$

The Cartan matrix is

$$C(A) = \begin{pmatrix} -\frac{1}{2}\frac{|\mathbf{x}_t, \mathbf{x}_{ttt}|}{|\mathbf{x}_t, \mathbf{x}_{tt}|} & 1 \\ -\frac{|\mathbf{x}_{tt}, \mathbf{x}_{ttt}|}{|\mathbf{x}_t, \mathbf{x}_{tt}|} & +\frac{1}{2}\frac{|\mathbf{x}_t, \mathbf{x}_{ttt}|}{|\mathbf{x}_t, \mathbf{x}_{tt}|} \end{pmatrix}$$

As it stands this matrix is not invariant because the parameter t is not invariant. However, we can change the parameter so that the diagonal elements vanish. Since this vanishing is an invariant condition, the new parameter τ is invariant, and thus the rest of the elements of A also become invariant. The diagonal elements will vanish if

$$|\mathbf{x}_\tau, \mathbf{x}_{\tau\tau\tau}| = |\mathbf{x}_\tau, \mathbf{x}_{\tau\tau}|_\tau = 0$$

or

$$|\mathbf{x}_\tau, \mathbf{x}_{\tau\tau}| = 1$$

Transforming from the old parameter to the invariant one we have

$$|\mathbf{x}_t, \mathbf{x}_{tt}| = |\mathbf{x}_\tau, \mathbf{x}_{\tau\tau}|\tau_t^3 = \tau_t^3$$

from which the invariant parameter, the *affine arclength* is obtained:

$$\tau = \int_{t_0}^{t} \mathrm{abs}|\mathbf{x}_t, \mathbf{x}_{tt}|^{1/3} dt$$

With this invariant parameter, there is only one significant element left in the Cartan matrix, namely the *affine curvature*

$$\kappa_a(\tau) = |\mathbf{x}_{\tau\tau}, \mathbf{x}_{\tau\tau\tau}|$$

Expressing this with the original parameter we see that the fourth derivative is needed:

$$\begin{aligned}\kappa_a(t) \;=\; & -\frac{5}{9}|\mathbf{x}_t, \mathbf{x}_{tt}|^{-8/3}|\mathbf{x}_t, \mathbf{x}_{ttt}|^2 \\ & +\frac{4}{3}|\mathbf{x}_t, \mathbf{x}_{tt}|^{-5/3}|\mathbf{x}_{tt}, \mathbf{x}_{ttt}| \\ & +\frac{1}{3}|\mathbf{x}_t, \mathbf{x}_{tt}|^{-5/3}|\mathbf{x}_t, \mathbf{x}_{tttt}|\end{aligned}$$

Following the general theorem of differential invariants cited earlier, the invariants τ, κ_a are a complete set that determines the curve up to a unimodular affine transformation.

Regarding a general affine transformation, these invariants still leave two undetermined coefficients: the starting point t_0 and the determinant of the affine transformation. If we plot $\kappa_a^{-1/2}$ with respect to τ we will obtain a curve which is affine invariant except for its starting point and its scale, which would affect both axes in the same way.

It is interesting to observe that the affine invariants $\tau, \kappa_a(\tau)$ are expressed as simple determinants. Thus their invariance can be verified directly, without relying on the Cartan method [3]. The same is true for the Euclidean curvature and for the algebraic projective invariants (but not for the differential projective ones). This is not surprising since the transformation of determinants involves only the Jacobian. However, the advantage of Cartan's method is that it can be used for higher dimensions and other transformation groups.

Since the method requires differentiation with respect to a parameter, it is not optimal in the sense that high derivatives are required to eliminate the parameter. The parameterless method described later should reduce the number of quantities that are needed to be extracted from the image.

5 Differential Invariants without Derivatives

We describe here a method that combines the locality of the differential invariants with the advantages of the algebraic method, such as avoiding the need for curve parametrization.

A problem with the differential method, as mentioned before, is that the arbitrary parameter is a source of ambiguity whose elimination requires higher derivatives then the projectivity itself needs. While the projectivity has only eight parameters that have to be eliminated, requiring the fourth derivatives, the unknown parameter forces us to obtain more information from the image, or higher derivatives, to eliminate it. On the other hand, the parameter is not in fact part of the geometry of the curve. The coordinates x, y of each point are sufficient to characterize the curve and the parameter is introduced artificially for convenience. The information contained in a derivative is partly wasted from a purely geometrical point of view because it informs us about the relation of x, y to the parameter rather than to each other, and to obtain the full geometrical information we are forced to go to higher derivatives. It would thus be profitable to find other quantities that are purely geometrical and so can be fully utilized for deriving geometrical invariants.

A way to approach the problem is to deal with an implicit representation of the curve, i.e. one of the form $f(x, y) = 0$, without an explicit parameter. The given curve itself is quite arbitrary and it is hard to find an f that will represent it in this way. However, at each point of it, one can find a simpler, osculating curve that can be represented implicitly and whose invariants are relatively easy to find.

An osculating curve is a generalization of the tangent. A tangent is a line having at least two points in common with the curve in an infinitesimal neighborhood, i.e. two "points of contact". This can be expressed as a condition on the first derivative. Similarly, a higher order osculating curve has more (independent) points of contact, and the condition on the derivatives can be written as

$$\frac{d^k}{dt^k}(f^*(x, y) - f(x, y)) = 0, \qquad (k = 0 \ldots n) \qquad (5)$$

with f^* being the osculating curve, f the given curve, and n the order of the osculation. Since the derivatives vanish, this condition is invariant to the parameter t. Since it has a geometric interpretation with points of contact, the condition is also projectively invariant.

To find invariants at a point of a given curve, we can find an algebraic (implicit) curve that osculates it at that point. The coefficients of this osculating curve are independent of any parametrization and we only need ten of them to find two independent invariants. (This includes the coordinates of the point itself which may already be known.) Thus the reliability of the method should be greater than the derivative based method which needs 18 quantities.

Obvious questions arise as to which algebraic curve we should choose as an osculating curve and how to calculate it. We solve this problem by a method similar to the one used for obtaining robust derivatives, i.e. we choose

a curve that preserves the first moments of the original data over some window.

Unlike other moment methods, ours is *local*, being applied over a neighborhood around each point. This method of local moment preservation is analyzed in detail in [14] and summarized in the next section, for the case of ordinary derivatives. We show both analytically and numerically that it leads to much greater reliability than other methods. We derive a quantitative relation between the various parameters involved, i.e. the the window size, the order of the moments to be preserved and the accuracy of the method. The analysis can be easily carried over to the implicit case. The equality of the first local moments also ensures that the curve f^* is indeed osculating to f.

Semi-differential invariants

One can reduce the number of data quantities needed at each point if some other information about the shape is known, such as the appearance of straight lines or known feature points. For example, a silhouette of an airplane can contain both curved parts and straight lines. Invariants involving both derivatives and reference points were found by [1] and [12].

The "derivativeless" method described above is perfectly suited for this situation, and again leads to saving in the number of data quantities needed from the image and imcreased reliability. If we have a known "feature" point (not necessarily on the curve), then for each point on the curve we can construct an osculating curve that touches it at fewer contact points, but will also pass through the known feature point. This is analogous to reducing the order of derivatives in the parametrized methods. Similarly, if we have a known line, we can demand that the osculating curve be tangent to this line. The invariants can be found as before. Of course, the correspondence between the appropriate parts of the shape has to be known.

6 Noise Resistant Derivatives

As we saw, the success of the differential method depends critically on a reliable differentiation method. In this section we develop such a method; our results have significant uses beside invariants. We analyze the requirements that a differentiation method has to meet, show that the Gaussian is not good enough for our purposes and present the method that was successful in our experiments. More details can be found in [14] and [10].

Two basic, and conflicting, requirements are involved: a) Accuracy: At least in the noiseless case, one would like to obtain the correct derivatives, at least at low orders. b) Smoothing, to alleviate the effect of noise and discretization. Generally, smoothing reduces the accuracy even in the analytic case as the more rapid changes in the function are smoothed out. The goal in designing a derivative filter is then to strike the correct balance between accuracy and smoothing. In this section we describe how this balance can be achieved.

As an illustration, we first show that the Gaussian gives the wrong result even for the simplest functions. Smoothing over x^2 gives

$$g(x,\sigma) \otimes x^2 = x^2 + \sigma^2$$

where $g(x,\sigma)$ is the Gaussian, and $\otimes$ means convolution. Similar results are obtained for derivatives. We can see that an error is introduced that increases as we increase the smoothing, i.e. higher σ. Similar results are obtained for higher powers or for taking derivatives. For effective smoothing σ should not be too small so this error can be substantial. This is a *systematic "oversmoothing error"*, as opposed the random noise that we want to smooth. As noted before, we do not expect a smoothing operator to give accurate results but we do want to improve the balance between accuracy and smoothing so that at least the simple, smooth functions will remain accurate.

We will now discuss this noise versus signal problem in terms of the ratio between the smoothing parameter σ, and a "natural" scale of the shape, s_0. For a smooth shape $f(x)$, we estimate this s_0 as the scale over which the relative change in the shape, $\Delta f/f$, is of the order of magnitude of 1. We then rescale the x axis so that $x_1 = x/s_0$, and rewrite the shape as $\bar{f}(x_1)$. For example, if $f(x) = \sin(x/s_0)$ then $\bar{f}(x_1) = \sin x_1$. This makes the derivatives of $\bar{f}$ with respect to x_1 of the order of magnitude of 1. In this way we "normalize" the signal in the x direction and deal with its scale separately.

A natural tool in dealing with the derivative in some neighborhood of a smooth shape is the Taylor expansion

$$f(x) = \bar{f}(\frac{x}{s_0}) = \sum_\nu \frac{\bar{f}^{(\nu)}}{\nu!}(\frac{x}{s_0})^\nu$$

with the derivatives $\bar{f}^{(\nu)} = \frac{d^\nu \bar{f}}{d(x/s_0)^\nu} \approx O(1)$. We will look at the result of Gaussian filtering at $x = 0$. It is easy to show that the error introduced by this smoothing is

$$g \otimes f - f \approx \bar{f}''(\frac{\sigma}{s_0})^2 + \frac{1}{8}\bar{f}^{(4)}(\frac{\sigma}{s_0})^4 + \ldots$$

If we want accurate results, we have to keep this error small. Looking at the leading term, we see that the accuracy is proportional to σ^2/s_0^2, so we have to keep σ small. Unfortunately, this limits the ability of the filter to smooth out the noise. A similar result is obtained for derivatives.

A way to improve the situation is to eliminate the leading terms in the expansions of the errors above. If the first term is eliminated, for instance, then the error will be reduced to $\approx \frac{1}{8}(\frac{\sigma}{s_0})^4$. This way we can obtain a much better accuracy for the same smoothing parameter σ as before (for $\sigma \leq s_0$). Alternatively, we can increase the smoothing without compromising accuracy.

Since the first error terms come from the first powers in the Taylor expansion of f, we need a filter F_l that will preserve the first l powers x^l:

$$F_l \otimes x^n = x^n, \qquad n = 0 \ldots l$$

(For the Gaussian $l = 1$.) It can be easily shown that the powers will be preserved if the first l moments of the filter vanish (and the 0th moment is normalized to 1). Defining the "normalized moments" as

$$m_n = \int (\frac{x}{\sigma})^n F_l(x) dx$$

where σ is now a measure of the filter size, e.g. the variance, we have the conditions

$$m_0 = 1; \qquad m_n = 0 \quad \text{for} \quad n = 1 \ldots l$$

The error in the smoothing of $f^{(k)}$ at $x = 0$ is now simply the $l+1$ term:

$$F_l \otimes f^{(k)} - f^{(k)} \approx \frac{m_{l+1}}{(l+1)!} \bar{f}^{(l+k+1)} (\frac{\sigma}{s_0})^{l+1} + \ldots$$

Thus, increasing the order l of the filter eliminates the powers $(\frac{\sigma}{s_0})^n$ up to $n = l$ and improves the accuracy to the above tolerance.

Accuracy Criterion

For good accuracy, the above error has to be small. This leads to an "accuracy criterion" for choosing appropriate parameters. Since $\bar{f}^{(\nu)} \approx O(1)$, we obtain the condition

$$\frac{m_{l+1}}{(l+1)!} (\frac{\sigma}{s_0})^{l+1} << 1 \qquad (6)$$

This criterion can be used to estimate the parameters in several ways. Given the meaningful scale of change of the signal, s_0, and the smoothing parameter σ, we can calculate the order l of the filter needed to lower the oversmoothing error to some acceptable level. σ can be determined from the estimated noise. Conversely, for a given order l, we can calculate the largest smoothing parameter σ that will still maintain a desired accuracy. Generally speaking, at low orders we need $\sigma < s_0$, but at high orders we can afford to have the smoothing σ bigger than s_0 because the factor $m_{l+1}/(l+1)!$ in eq. (6) is small.

One way of obtaining such zero-moment filters is by multiplying a Gaussian with appropriate polynomials. The filter of order l, eliminating errors up order l, is (on an infinite interval)

$$F_l = \sum_{i=0}^{l} (a_i P_i(x)) g(x)$$

where $P_i(x)$ are Hermite polynomials which are orthogonal with respect to the Gaussian weight function. The coefficients a_i are chosen so that the first l moments of the filter vanish, in accordance with our conditions.

Oversmoothing is only one problem with the Gaussian. The truncation of the infinite filter is also a serious problem, because the derivative $g^{(n)}$ becomes meaningless at the ends of the window and contributes a large error [14]. Discrete versions of this method on a finite window are described in detail in [10]. Closed form filters are derived there that yield differentiation filters of the desired orders. We use here one of these methods, based on the Krawtchouk polynomials. They are defined by the condition of orthogonality with respect to the binomial weight function.

In summary, we have been able to obtain good estimation of high derivatives by increasing the filter size σ and its order l to meet the above accuracy criterion.

7 Experiments

We describe here experiments done on synthetic shapes. Despite the limitations of such experiments, we were able to obtain a good idea of the reliability of the high derivatives by perturbing the data and looking at the change in the invariants. The results, with the differentiation method that we employed, were very good.

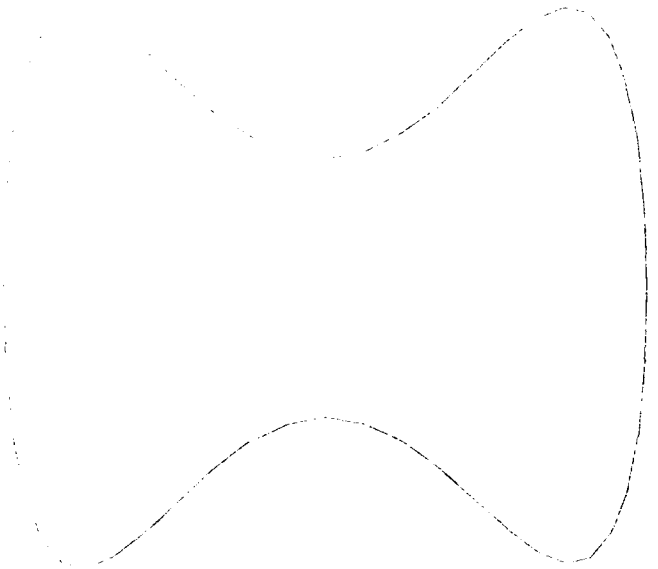


Figure 1: Peanut: Projection 1

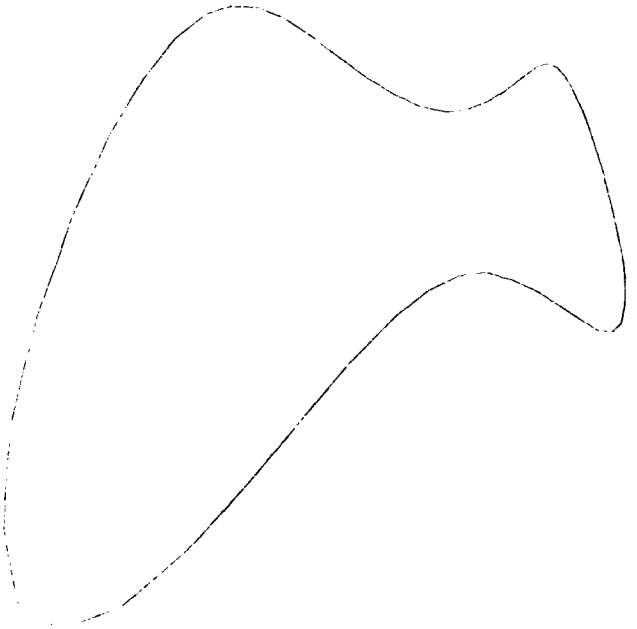


Figure 2: Peanut: Projection 2

Projective Invariants

Fig. 1 shows a "peanut" shape, created as

$$\begin{aligned} x(t) &= 2\cos t \\ y(t) &= \sin t + \frac{1}{2}\sin 3t \end{aligned}$$

and Fig. 2 is its projection with tilt and slant of about 45°. Both projections were discretized and the derivatives were calculated from the discrete images.

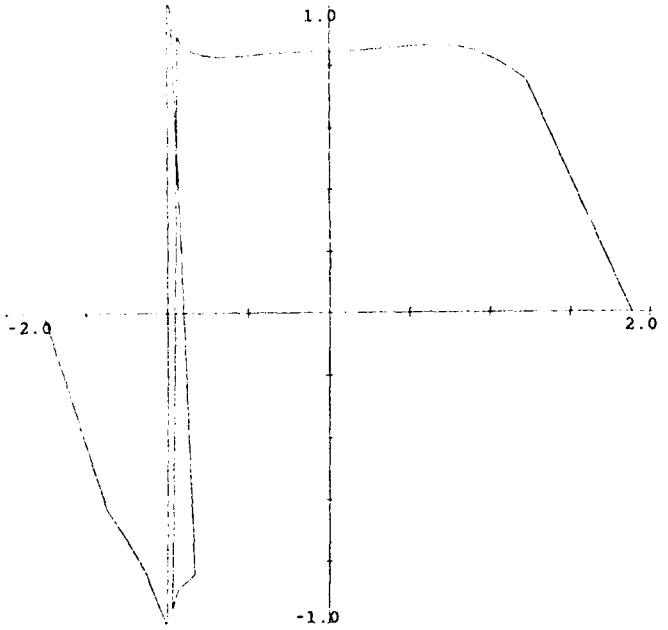


Figure 3: Peanut: Invariant curve

To suppress noise in the derivatives, we would like the smoothing factor of the filter σ to be big. But, according to the analysis of the last section, σ cannot be much bigger than the scale of variation of the shape, which here is $s_0 = 1/3$. We chose the order of the filter as $l = 10$, and found that very good accuracy is obtained with $\sigma = 0.4$, in line with our accuracy criterion, eq. (6). With these parameters, we obtained very reliable and noise resistant invariants. Perturbing the data by 10% yielded perturbations in the invariants of less than that amount, without sacrifying the distinctiveness of the curve. A smaller σ would require a smaller window but would result in more sensitivity to noise. This shows that there is no serious problem in using high derivatives as long as the differentiation method and its parameters are chosen properly.

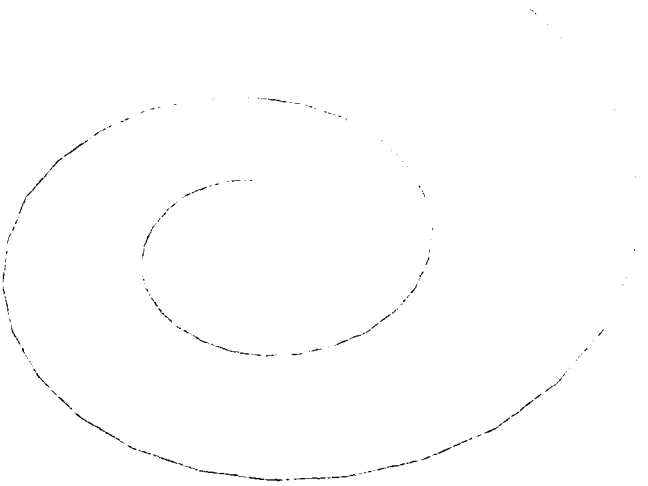
In Fig. 3 we have plotted normalized versions of our modified semi-invariants, $\hat{I}_1, \hat{I}_2$, which were similar for both projections, for a quarter of the images.

$$I_1 = \frac{P_2}{|P_2|}\sqrt{|P_2|}, \qquad I_2 = \sqrt[3]{P_3^*}$$

$$\hat{I}_1 = \frac{I_1}{\sqrt{1+I_2^2}} \qquad \hat{I}_2 = \frac{I_2}{\sqrt{1+I_2^2}}$$

The first line above makes the invariants of equal weight. The second one is a normalization done to counter the effect of the singularity that we observe. This singularity comes from the inflection point of the shape, i.e. where the curvature has a zero crossing. At this point both invariants tend to infinity and are equal in absolute magnitude, but I_2 jumps between $\pm\infty$. The normalization reduces the infinities to ± 1.

Inflection points are themselves projective invariants and can be found in other ways as well. Thus their location on the curve is fixed and should not pose a problem for our method. The power of the differential invariants is in providing information about the behavior of the curve between the inflection points. We can see that the location of the inflection points is determined quite precisely in this method, an observation that has significance for the general problem of accurate edge detection.


Figure 4: Spiral: Projection 1

For the Krawtchouk filter that we have used, we have $\sigma \approx w/l$, with w being the width of the window. Thus we needed a window of width $l\sigma = 4$, which is more than half the shape (but we only used 11 points in each window). We believe that an optimized differentiation method such as proposed in [14] can provide the same smoothing with a smaller window.

The same method was applied to logarithmic spirals. Figs. 4 and 5 show the two projections. An interesting result, which can be proven analytically, is that a logarithmic spiral has constant projective invariants along its arclength. This may be interesting in connection with recognizing biological forms because according to some theories [5] many such forms can be segmented into spiral parts.

Affine Invariants

The affine invariants developed before are invariant to reparametrization of the curve as well as the projection so we could test this property under conditions of discretization. The derivatives were calculated in the same way as before, and again the fourth derivative was needed. Fig. 6 shows an affine projection of our peanut with a unimodular 2×2 matrix. Fig. 7 shows the invariant curves for both projections, with different parametrizations. In both curves the affine invariant $\kappa_a^{-1/2}$ is plotted against the affine arclength τ. The discrete points are also shown.

We can see that the curves are quite close. The biggest difference is in the region where the affine curvature is at

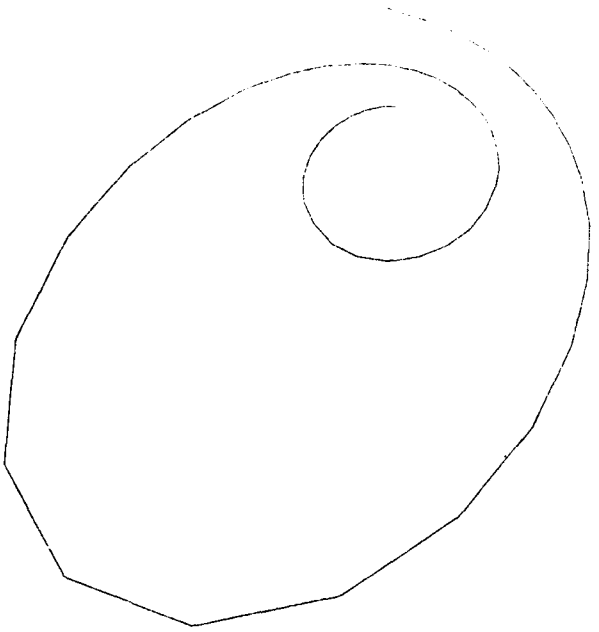


Figure 5: Spiral: Projection 2

its maximum, at which point it also changes sign. This region surrounds a singularity so the results there are somewhat less reliable but still very informative. The point at which the affine curvature touches zero without changing sign is the inflection point, i.e. the Euclidean curvature changes sign there.

We note that unlike the projective case, here we have to use the same starting point t_0, otherwise the curves will be shifted horizontally. Also, for a general affine transformation (with arbitrary determinant) the scale of the invariant curve will change. We chose the invariants so that the scale change will be the same for both axes.

8 Conclusions

We have developed new methods of obtaining differential invariants and implemented some of them experimentally. We have obtained an invariant signature that can be used to recognize a plane curve regardless of the point of view from which the curve is seen. We have found a new method of smoothing and differentiation and showed that it is robust to noise, which makes the signature reasonably reliable. Our smoothing method, involving preservation of moments in local neighborhoods, has significant applications in other fields as well.

References

[1] Barrett, E., Payton, P., Haag, N. and Brill, M. [1991], "General methods for determining projective invariants in imagery", *CVGIP:IU* **53**, 45–65.

[2] Brown, C.M. [1991] "Numerical evaluation of differential and semi-differential invariants", *TR-393*, University of Rochester Computer Science Department, September 1991.

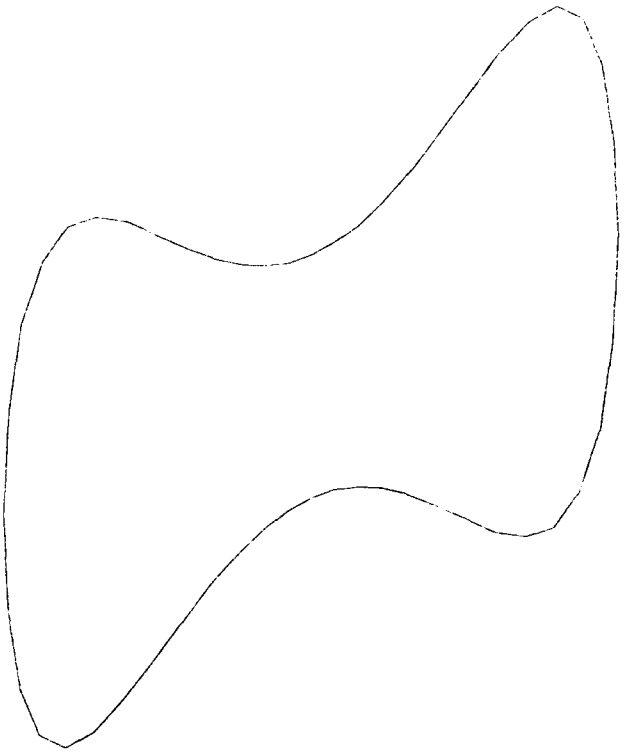


Figure 6: Peanut: Projection 3

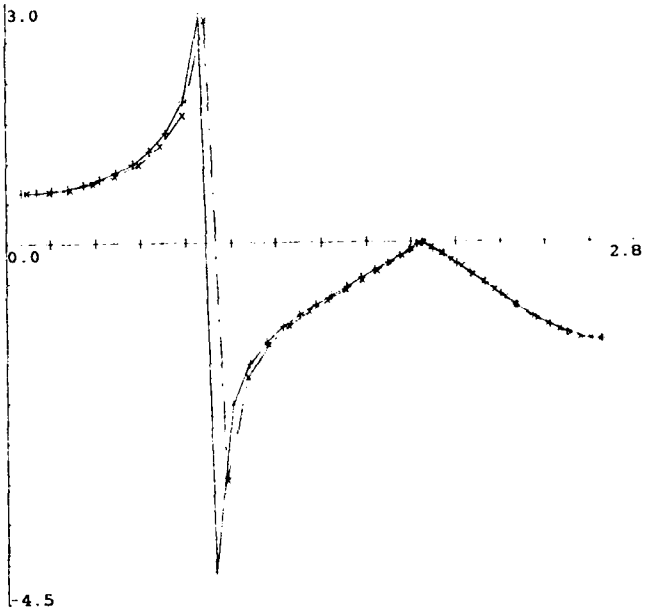


Figure 7: Peanut: Affine invariant curves

[3] Bruckstein, A. and Netravali, A. [1990], "On differential invariants of planar curves and recognizing partially occluded planar objects", AT&T TR, July 1990.

[4] Burns, J.B., Weiss, R., and Riseman, E.M. [1990], "View variation of point set and line segment features", *Proc. DARPA IU Workshop*, 650–659.

[5] Cook, T. [1914] *The Curves of Life*, New York: Dover.

[6] DARPA-ESPRIT Workshop on Invariance, Proceedings [1991], Reykjavik, Iceland, March 1991.

[7] Duda, R.O. and Hart, P.E. [1973] *Pattern Recognition and Scene Analysis*, New York: Wiley.

[8] Forsyth, D., Mundy, J.L., Zisserman, A., and Brown, C.M. [1990], "Invariance—A new framework for vision", *Proc. ICCV*, 598–605.

[9] Guggenheimer, H. [1963] *Differential Geometry*, New York: Dover.

[10] Meer, P. and Weiss, I. [1989], "Smoothed differentiation filters for images", Center for Automation Research, University of Maryland, TR 424.

[11] Nielsen, L. and Sparr, G. [1991] "Projective area invariants as an extension of the cross ratio", *CVGIP:IU* **54**, 145–159.

[12] Van Gool, L., Wagemans, J., Vandeneede, J., Oosterlinck, A., [1990], "Similarity extraction and modeling", *Proc. ICCV*, 530–534.

[13] Weiss, I. [1988] "Projective invariants of shapes", *Proc. DARPA IU Workshop*, 1125–1134.

[14] Weiss, I. [1991] "High order differentiation filters that work", University of Maryland, Center for Automation Research, TR CAR 545.

[15] Weiss, I., Meer, P. and Dunn, S.M. [1991], "Robustness of algebraic invariants", *Proc. DARPA-ESPRIT Workshop on Invariance*, Reykjavik.

[16] Wilczynski, E.J. [1906], *Projective Differential Geometry of Curves and Ruled Surfaces*, Leipzig: Teubner.

Polynomial-Time Object Recognition in the Presence of Clutter, Occlusion, and Uncertainty

Todd A. Cass
Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Abstract

We consider the problem of object recognition via local geometric feature matching in the presence of sensor uncertainty, occlusion, and clutter. We present a general formulation of the problem and a polynomial-time algorithm which guarantees finding all feasible interpretations of the data, modulo uncertainty, in terms of the model. This formulation applies to problems involving both 2D and 3D objects.

We will describe the theory in general, and analyse particular cases in detail including an approach achieving efficient practical performance. Algorithms, implementations, and experiments are presented.

1 Introduction

The task considered here is the recognition by computer of a known object in some environment via the use of sensory data, such as a visible light image, derived from the environment. Object models and sensory data are represented as sets of local geometric features, e.g. points and lines. The problem is formulated as matching model features and data features to determine the pose of an instance of the model. This problem is hard because there are spurious and missing features, as well as sensor uncertainty. This paper presents improvements and extensions to earlier work[9, 7] describing robust, complete, and provably correct methods for polynomial-time object recognition in the presence of clutter, occlusion, and sensor uncertainty.

We assume the uncertainty in the sensor measurements of the data features is bounded. A model pose is considered feasible for a given model and data feature match if at that pose the two matched features are aligned modulo uncertainty, that is, if the image of the model feature falls within the uncertainty bounds of the data feature. We show that, given a set of model and data features and assuming bounded sensor uncertainty, there are only a polynomial number of qualitatively distinct poses matching the model to the data. Two different poses are qualitatively distinct if the sets of feature matches aligned (modulo uncertainty) by them are different. The approach is based on determining the set of qualitatively different poses. We call this approach *pose equivalence analysis*. A previous paper [8] introduced the idea of pose equivalence analysis; this paper contributes a simpler explanation of the approach based on linear constraints and transformations, outlining the general approach for the case of 3D and 2D models with 2D data. This formalization provides a simple and clean mathematical framework within which to analyse the feature matching problem in the presence of bounded geometric uncertainty, providing insight into the fundamental nature of this type of feature matching problem. We analyze a particular case of 2D models and planar transformations to illustrate how the structure of the matching problem can be exploited to develop efficient matching algorithms based on this approach.

1.1 Robust Object Localisation via Feature Matching

Object *localization* will be defined as the problem of determining the geometric correspondence between the model and some a priori unknown subset of the data. We're given a geometrical model of the spatial structure of an object and the problem is to select data subsets corresponding to instances of the model, and determine the position and orientation, or *pose*[1], of the object in the environment by matching the geometric model with instances of the object represented in the sensory data.

In this paper the model and the sensory data will be represented in terms of *local geometric features* consisting of points or lines and possibly curve normals. Localisation is achieved via matching model and data feature subsets. There are four features of this task domain which are important to consider in the work: The feature matching problem is difficult because the correct feature correspondences are unknown; there are spurious data features; there can be model features missing from the data due to occlusion and failures in feature extraction; most importantly, the sensory data are subject to geometrical uncertainty that greatly affects the process of robust and correct detection and pose determination.

Localisation is reasonably divided into a pose hypothesis stage and a pose verification stage. This paper con-

[1] The pose of the model is its position and orientation, which is equivalent to the transformation producing it. In this paper *pose* and *transformation* will be used interchangeably.

siders pose hypothesis construction via feature matching.

1.2 Robustness, Completeness, and Tractability

There are three important criteria by which to analyse methods for object localisation: *robustness*, *completeness*, and *tractability*. The robustness requirement means that careful attention is paid to the geometric uncertainty in the features, so no correct feature correspondences are missed due to sensor error. The completeness requirement means that *all* sets of geometrically consistent feature correspondences are found, including the correct ones. The tractability requirement simply means that a polynomial-time and hopefully efficient algorithm exist for the matching procedure. Except for our previous work [7, 8], and recent work by Breuel[4], among those existing methods accurately accounting for uncertainty, none can both guarantee that all feasible object poses will be found and do so in polynomial time. Those that do account for error and guarantee completeness have expected-case exponential complexity[13].

Uncertainty is the main factor which makes the localisation problem difficult. If there is no uncertainty then simple polynomial-time algorithms are possible for localisation, guaranteeing success[15]. However, if the measured position of features are used without accounting for possible deviation from the correct positions then these approaches cannot guarantee the correct matching of the model to the data.

1.3 Correspondence Space vs. Pose Space

For given sets of model and data features, we defined localisation as both determining which subset of data features correspond to the model features, and how they correspond geometrically. Localisation can be accomplished by either searching for geometrically consistent feature correspondences or searching for the model transformation or pose geometrically aligning model and data features. Techniques based on these approaches can be called *correspondence space* methods and *pose space* methods, respectively.

Correspondence space is the power set of the set of all model and data feature pairs: $2^{\{m_i\}\times\{d_j\}}$ where $\{m_i\}$ and $\{d_j\}$ represent the sets of model and data features, respectively. We define a *match set* $\mathcal{M} \in 2^{\{m_i\}\times\{d_j\}}$ as an arbitrary set of model and data feature matches. Localisation can be achieved by finding geometrically consistent match sets, that is, match sets for which there exists some pose of the model aligning modulo uncertainty the matched features. One way to structure this is as a search through the correspondence space, often structured as a tree search[14, 3]. Correspondence space is an exponential-sized set, and although enforcing geometric consistency in the search prunes away large portions of the search space, it has been shown that the expected search time is still exponential[13].

Pose or transformation space is the space of possible transformations on the model. Localisation can be accomplished by searching pose space for model poses aligning modulo uncertainty model features to data features. Examples of techniques searching pose space are pose clustering[19], transformation sampling[6, 5], and the method described in this paper, pose equivalence analysis[8]. The pose space is a high dimensional, continuous space, and the effects of data uncertainty and missing and spurious features make effectively searching it to find consistent match sets difficult.

The approach described in this paper provides a framework for unifying the correspondence space approach and the pose space approach.

2 Pose Equivalence Analysis

We will represent the model and the image data in terms of local geometric features such as points and line segments derived from the object's boundary, to which we may also associate an orientation. Denote a model feature $m = (\mathbf{p}_m, \hat{\theta}_m)$ by an ordered pair of vectors representing the feature's position and orientation, respectively. Similarly the measured geometry of a data feature is given by $d = (\mathbf{p}_d, \hat{\theta}_d)$. Define U_i^p and U_i^θ to be the uncertainty region for the position and orientation, respectively, for data feature d_i. We assume that the uncertainty in position or orientation are independent. The true position of d_i falls in the set U_i^p, and its true orientation falls in the set U_i^θ. We can think of U_i as a region surrounding d_i. For most of the following analysis we will consider the case where the features are simply points in the plane without an associated orientation; we then show how to incorporate orientation constraints.

Correctly hypothesising the pose of the object is equivalent to finding a transformation on the model into the scene aligning the model with its instance in the data, by aligning individual model and data features. *Aligning* a model feature and a data feature consists of transforming the model feature such that the transformed model feature falls within the geometric uncertainty region for the data feature. We can think of the data as a set of points and uncertainty regions $\{(\mathbf{p}_{d_j}, U_j^p)\}$ in the plane, where each measured data position is surrounded by some positional uncertainty region U_j^p. A model feature with position $\mathbf{p}_{m_i}$ and a data feature with position $\mathbf{p}_{d_j}$ are aligned via a transformation T if $T[\mathbf{p}_{m_i}] \in U_j^p$. Because the pose of an instance of the model in the environment and the model itself are related by a rigid transformation, some transformations will align many correct model and data feature matches although which matches are correct is unknown a priori. Intuitively, the whole problem is then to find single transformations simultaneously aligning in this sense a large number of pairs of model and image features.

One of the main contributions of this work, and the key insight of this approach is the idea that under the bounded uncertainty model there are only a polynomial number of *qualitatively different* transformations or poses aligning subsets of a given model feature set with subsets of a given data feature set. Finding these *equivalence classes* of transformations is equivalent to finding all qualitatively different sets of feature correspondences. Thus we need not search through an exponential number of sets of possible feature correspondences as previ-

ous systems have, nor consider an infinite set of possible transformations.

In the 2D case the transformations will consist of a planar rotation, scaling, and translation. We've said that a model feature m_i and data feature d_j are *aligned* by a transformation T iff $T[m_i] \in U_j$. Two transformations are *qualitatively similar* if and only if they align in this sense exactly the same set of feature matches. All transformations which align the same set of feature matches are *equivalent*. Thus there are equivalence classes of transformations. This is a key insight. More formally, let Ω be the transformation parameter space, and let $T \in \Omega$ be a transform. Define $\varphi(T) = \{(m_i, d_j) | T[m_i] \in U_j\}$ to be the set of matches aligned by the transformation T. The function $\varphi(T)$ partitions Ω forming equivalence classes of transformations E_k, where $\Omega = \bigcup_k E_k$ and $T \equiv T' \iff \varphi(T) = \varphi(T')$. The entire recognition approach developed in this paper is based upon computing these equivalence classes of transformations, and the set of feature matches associated with each of them.

2.1 Relating Pose Space and Correspondence Space

If a model feature m_i and a data feature d_j are to correspond to one another, the set of transformations on the model feature which are *feasible* is given by the set of transformations $\mathcal{F}_{m_i,d_j} = \{T \in \Omega | T[m_i] \in U_j\}$. Let $\mathcal{M} = \{(m_i, d_j)\}$ be some *match set.*[2]. A match set is called *geometrically consistent* iff $\bigcap_{(m_i,d_j)\in\mathcal{M}} \mathcal{F}_{m_i,d_j} \neq \emptyset$, that is iff there exists some transformation which is feasible for all $(m_i, d_j) \in \mathcal{M}$.

The match set given by $\varphi(T)$ for some transformation T is called a *maximal geometrically-consistent match set.* A match set $\mathcal{M}$ is a maximal geometrically-consistent match set (or, a *maximal match set*) if it is the largest geometrically consistent match set at some transformation T. Thus by definition the match set given by $\varphi(T)$ is a maximal match set. The function $\varphi(T)$ is a mapping from transformation space to correspondence space. $\varphi(T) : \Omega \longrightarrow 2^{\{m_i\}\times\{d_j\}}$, and there is a one-to-one correspondence between the pose equivalence classes and the maximal match sets given by $\varphi(T) \longleftrightarrow E_k$ iff $T \in E_k$. The function $\varphi(T)$ partitions the infinite set of possible object poses into a polynomial-sized set of pose equivalence classes; and identifies a polynomial sized subset of the exponential-sized set of possible match sets.

The important point is that the pose equivalence classes and their associated maximal match sets are the only objects of interest: all poses within a pose equivalence class are qualitatively the same; and the maximal geometrically consistent match sets are essentially the only sets of feature correspondences that need be considered because they correspond to the pose equivalence classes. Figure 1 shows the relationship between correspondence space and transformation space. Note

[2]This is also sometimes called a correspondence, or a matching. To clarify terms, we will define a *match* as a pair of a model feature and a data feature, and a *match set* as a set of matches. The term *matching* implies a match set in which the model and data features are in one-to-one correspondence.

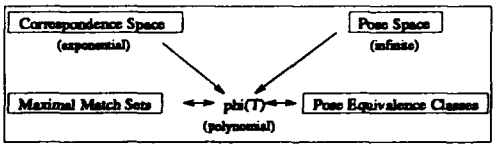


Figure 1: The pose space is partitioned into a polynomial number of pose equivalence classes, each associated with a maximal match set, via the function $\varphi(T)$. The maximal match sets are the only sets of feature correspondences we need to consider.

that this implies we do not need to consider *all* consistent match sets, or search for one-to-one feature matchings, because they are simply subsets of some maximal match set, and provide no new pose equivalence classes. However, given a match set we can easily construct a maximal, one-to-one matching between data and model features[16].

One distinction between this approach, which works in transformation space, and robust and complete correspondence space tree searches[14, 3] is that for each maximal geometrically consistent match set (or equivalently for each equivalence class of transformations) there is an exponential sized set (in terms of the cardinality of the match set) of different subsets of feature correspondences which all specify the same set of feasible transformations. Thus the straightforward pruned tree search does too much (exponentially more) work by exploring these subsets of maximal match sets. This is part of the reason why these correspondence space search techniques have exponential expected case performance, yet our approach is polynomial (see section 8 and the discussion of Breuel's[4] work).

2.2 Feature Matching Requires Only Polynomial Time

Formalising the localization problem in terms of bounded uncertainty regions and transformation equivalence classes allows us to show that it can be solved in time polynomial in the size of the feature sets. Cass[7] originally demonstrated this using quadratic uncertainty constraints. This idea can be easily illustrated using the linear vector space of 2D scaled rotations and translations[2, 3], and the linear constraint formulation used by Baird[3] and recently by Breuel[4]. In the 2D case the transformations will consist of a planar rotation, scaling, and translation. Any vector $\mathbf{s} = [s_1, s_2]^T = [\sigma\cos\theta, \sigma\sin\theta]^T$ is equivalent to a linear operator $\mathbf{S}$ performing a rigid rotation by an orthogonal matrix $\mathbf{R} \in SO_2$ and a scaling by a factor $\sigma \in \mathbb{R}$, $0 < \sigma < \infty$, where $\mathbf{S} = \sigma\mathbf{R} = \sigma \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} s_1 & -s_2 \\ s_2 & s_1 \end{bmatrix}$. We denote the group of all transformations by Ω, with translations given by $\mathbf{t} = [t_1, t_2]^T$ and scaled rotations given by $\mathbf{s} = [s_1, s_2]^T$, so a point $\mathbf{x}$ is transformed by $T[\mathbf{x}] = \mathbf{S}\mathbf{x} + \mathbf{t}$, and a transformation, T, can be represented by a vector $T \leftrightarrow [s_1, s_2, t_1, t_2]^T \in \mathbb{R}^4$. Thus the set of all scaled rigid planar motions is isomorphic to $\mathbb{R}^4$.

By assuming k-sided polygonal uncertainty regions[3] U_i^p and following the formulation of Baird, the uncertainty regions U_i^p can be described by the set of points $\mathbf{x}$ satisfying inequality (1): $(\mathbf{x}-\mathbf{p}_{d_j})^T\hat{\mathbf{n}}_l \leq \epsilon_l$ for $l = 1, ..., k$ and thus by substitution the set of feasible transformations for a feature match (m_i, d_j) are constrained by inequality (2): $(\mathbf{S}\mathbf{p}_{m_i} + \mathbf{t} - \mathbf{p}_{d_j})^T\hat{\mathbf{n}}_l \leq \epsilon_l$ for $l = 1, ..., k$ which can be rewritten as constraints on the transformation vector $[s_1, s_2, t_1, t_2]^T$ as $s_1(p^x_{m_i}n^x_l + p^y_{m_i}n^y_l) + s_2(p^x_{m_i}n^y_l - p^y_{m_i}n^x_l) + t_1n^x_l + t_2n^y_l \leq \mathbf{p}_{d_j}{}^T\hat{\mathbf{n}}_l + \epsilon_l$, $l = 1, ..., k$, where $\hat{\mathbf{n}}_l$ is the unit normal vector and ϵ_l the scalar distance describing each linear constraint for $l = 1, ..., k$. The first set of linear inequalities, (1), delineate the polygonal uncertainty regions U_j^p by the intersection of k halfplanes, and the second set of inequalities, (2), provides k hyperplane constraints in the linear transformation space Ω. The intersection of the k halfspaces forms the convex polytope $\mathcal{F}_{m_i,d_j}$ of feasible transformations for match (m_i, d_j). The arrangement[4] of these mn convex polytopes forms the partition of transformation space into equivalence classes. To see this another way, consider the set of k hyperplanes for each of the nm feature matches (m_i, d_j). The arrangement constructed by these kmn hyperplanes partitions the set of transformations into cells. It is well known from computational geometry that the complexity of the arrangement of kmn hyperplanes in $\mathbb{R}^4$ is $\Theta(k^4m^4n^4)$ in terms of the number of elements of the arrangement. These elements are called *k-faces* where a 0-face is a vertex, a 1-face is an edge, a 3-face is a facet, and a 4-face is a cell, and these elements can be constructed and enumerated in $O(k^4m^4n^4)$ time[10].

We are interested in the relationship between the equivalence classes E_k of transformations defined by the function $\varphi(T)$, and the elements of the arrangement imposed by the set of all kmn constraint hyperplanes. The transformation equivalence classes are the cells and faces formed by the arrangement of the mn convex polytopes $\mathcal{F}_{m_i,d_j}$ in the transformation space[7]. So the arrangement formed by the feasible regions $\mathcal{F}_{m_i,d_j}$ is in a sense a subset of the arrangement formed by the kmn hyperplanes, whose complexity is higher. Thus there are elements of the hyperplane arrangement which do not delineate equivalence classes. Intuitively, the cells defined by the arrangement of the kmn hyperplanes are a finer subdivision of Ω than that imposed by $\varphi(T)$. Thus the number of qualitatively different poses is bounded by $O(k^4m^4n^4)$.

To construct a provably correct and complete polynomial-time pose hypothesis algorithm for object localisation we enumerate the set of pose equivalence classes by deriving them from this arrangement induced by the two feature sets and the uncertainty constraints. Each equivalence class is associated with a geometrically consistent set of feature matches, and so we simply select those consistent match sets of significant size. This is a simple illustration that the problem of determining geometrically consistent feature correspondences in the 2D case can be done in polynomial time, and the solution is correct.

Good pose hypotheses are given by regions where the constraints on many feature matches are satisfied. We need some way of evaluating hypotheses (prior to verification) to determine how good they are. Intuitively we wish to know how much of the data is explained in terms of the model. Two possible measures are the size of the maximal match sets, and the size of the largest one-to-one matching contained in the maximal match set. We use an approximation to the later consisting of the minimum of the number of distinct image features and distinct model features[16].

2.3 The Case of 3D models and 2D data

Of particular interest is the localization of 3D objects from 2D image data. We'll consider the case where the transformation consists of rigid 3D motion, scaling and orthographic projection; where a model point $\mathbf{p}_{m_i}$ is transformed by $T[\mathbf{p}_{m_i}] = \mathbf{S}\mathbf{p}_{m_i} + \mathbf{t}$ with $\mathbf{S} = \begin{bmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \end{bmatrix} = \sigma\mathbf{P}\mathbf{R}$ and $\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$, $\mathbf{R} \in SO_3$, and $\sigma > 0 \in \mathbb{R}$. In the case of planar 3D objects this corresponds to the transformation $\mathbf{S} = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix}$, $s_{ij} \in \mathbb{R}$, describing the projection of all rotations and scalings of a planar 3D object. Baird[3] and Breuel[4] noted the linear formulation applies to any affine transformation. To exploit linear uncertainty constraints on the feasible transformations as before, we must have the property that the matrices $\mathbf{S}$ form a vector space. This is true for $\mathbf{S} = \begin{bmatrix} s_1 & -s_2 \\ s_2 & s_1 \end{bmatrix}$, $s_1, s_2 \in \mathbb{R}$, in the case of 2D models and 2D transformations; and the case of $\mathbf{S} = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix}$, $s_{ij} \in \mathbb{R}$, for planar models and 3D transformations, but is not the case for general 3D models with $\mathbf{S} = \sigma\mathbf{P}\mathbf{R}$ because the components of $\mathbf{S}$ must satisfy $\mathbf{S}\mathbf{S}^T = \sigma^2\mathbf{I}$, where $\mathbf{I}$ is the 2×2 identity matrix. To handle this case we follow the following strategy. We compute equivalence classes in an extended transformation space $\tilde{\Omega}$ which is a vector space containing the space Ω (see also [20]). After computing transformation equivalence classes we then restrict them back to the non-linear transformation space we are interested in. So consider the vector space $\tilde{\mathcal{S}}$ of 2×3 matrices $\tilde{\mathbf{S}} = \begin{bmatrix} \tilde{s}_{11} & \tilde{s}_{12} & \tilde{s}_{13} \\ \tilde{s}_{21} & \tilde{s}_{22} & \tilde{s}_{23} \end{bmatrix} \in \tilde{\mathcal{S}}$ where $\tilde{s}_{ij} \in \mathbb{R}$, and define $\tilde{\Omega}$ to be the set of transformations $(\tilde{\mathbf{S}}, \mathbf{t}) \in \tilde{\Omega}$ where $\tilde{T}[\mathbf{p}_{m_i}] = \tilde{\mathbf{S}}\mathbf{p}_{m_i} + \mathbf{t}$ as before. The set $\tilde{\Omega}$ is isomorphic to $\mathbb{R}^6$. Again expressing the uncertainty regions in the form of linear constraints we have $(\tilde{\mathbf{S}}\mathbf{p}_{m_i} + \mathbf{t} - \mathbf{p}_{d_j})^T\hat{\mathbf{n}}_l \leq \epsilon_l$ for $l = 1, ..., k$ or $(\sum \alpha_{pq}\tilde{s}_{pq}) + t_1n^x_l + t_2n^y_l \leq \mathbf{p}_{d_j}{}^T\hat{\mathbf{n}}_l + \epsilon_l$ for $l = 1, ..., k$, where the constants α_{pq} are functions of $\mathbf{p}_{m_i}$ and $\hat{\mathbf{n}}_l$. This describes k constraint hyperplanes in

[3] Each data feature d_j can have an arbitrary polygonal uncertainty region by considering a separate set $\{(\mathbf{n}_{l,j}, \epsilon_{l,j})\}$ of constraint parameters for each data feature.

[4] The computational geometric term is *arrangement* for the topological configuration of geometric objects like linear surfaces.

the linear, 8-dimensional transformation space $\tilde{\Omega}$. The kmn hyperplanes due to all feature matches again form an arrangement in $\tilde{\Omega}$. Analogous to the 2D case there are $\Theta(k^8m^8n^8)$ elements in this arrangement.

For the case of 3D planar objects under the transformation $T[\mathbf{p}_{m_i}] = \mathbf{S}\mathbf{p}_{m_i} + \mathbf{t}$ with $\mathbf{S} = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix}$, the transformation space is 6-dimensional and there are $\Theta(k^6m^6n^6)$ elements in the arrangement. Note that the special case of planar rotation, translation, and scaling is a restriction of this transformation to those matrices $\mathbf{S}$ satisfying $\mathbf{S}\mathbf{S}^T = \sigma^2\mathbf{I}$ which are the matrices of the form $\mathbf{S} = \begin{bmatrix} s_1 & -s_2 \\ s_2 & s_1 \end{bmatrix}$, $s_1, s_2 \in \mathbf{R}$, shown before.

Define $\tilde{\mathcal{F}}_{m_i,d_j} = \{\tilde{T} \in \tilde{\Omega} | \tilde{T}[\mathbf{p}_{m_i}] \in U_j^p\}$ and $\varphi(\tilde{T}) = \{(m_i, d_j); \tilde{T}[\mathbf{p}_{m_i}] \in U_j^p\}$, which as before defines equivalence classes of transformations $\tilde{E}_k$. The equivalence classes are the elements of the arrangement of the mn convex polytopes $\tilde{\mathcal{F}}_{m_i,d_j}$ which as described before is related to the arrangement of the constraint hyperplanes but of lower complexity, thus there are $O(k^8m^8n^8)$ equivalence classes $\tilde{E}_k$.

To consider the case where the general 2×3 linear transformation, $\tilde{\mathbf{S}}$ is restricted to the case, $\mathbf{S}$ of true 3D motion, scaling, and projection to the image plane, we simply eliminate elements of the arrangement which do not intersect the quadratic surface described by the constraints $\tilde{\mathbf{S}}\tilde{\mathbf{S}}^T = \sigma^2\mathbf{I}$. We still have $O(k^8m^8n^8)$ maximal match sets with the restricted transformation, although the equivalence classes are more complicated.

As was shown in [7] these same ideas apply to cases using non-linear uncertainty constraints, such as circles. The basic idea is the same however in these cases we must analyze an arrangement of quadratic surfaces which is computationally more difficult.

3 Efficiently Exploring Pose Equivalence Classes

We see from the previous analysis that there is only a polynomial sized set of qualitatively different model poses. A simple algorithm for object localisation consists of constructing the transformation equivalence classes, or representative points of them, from the arrangement of constraints in transformation space. For example a short-cut is to simply compute the vertices of the arrangement of the hyperplanes, rather than the entire arrangement. This is essentially the approach taken by Cass[7] where the constraints were quadratic. Every element of the arrangement is bounded by some vertex, thus by analysing the set of vertices we can form a set of representative transformations covering all equivalence classes.

Unfortunately this straightforward approach is impractical because of the complexity of computing either the arrangement or representatives of the elements of the arrangement. So the idea is to develop algorithms to explore the arrangement in an efficient way finding those regions of transformation space associated with large maximal match sets, i.e. transformation satisfying a large number of constraints, but without explicitly constructing or exploring the entire arrangement if possible. Developing efficient methods for exploring pose equivalence classes to find interesting ones now becomes the focus of algorithms based on this formalization; which is really a problem in computational geometry.

Because the underlying problem is that of pairwise matching the local features of a rigid object with fixed data features, there is considerable structure to the problem. First, when considering the formulation with linear constraints, we do not have an arbitrary arrangement of kmn hyperplanes in Ω, rather each feature match is associated with a convex polytope $\mathcal{F}_{m_i,d_j}$ in Ω describing the set of feasible transformations. The arrangement in Ω in which we are interested is the arrangement of these mn polytopes.

The main point regarding the structure of the constraints is that the complexity of the arrangement of feasible polytopes depends on how well the model matches the data. The more constraints that can be simultaneously satisfied, the higher the complexity of their arrangement is in transformation space. So the complexity cannot get arbitrarily close to the upper bound unless the model matches most of the data at some transformation, a case in which we can't resolve the data anyway because it is swamped in uncertainty. In typical cases of interest we would expect the model features to match a large number of data features in only a small number of poses, so the complexity of the arrangement is hopefully much less that the upper bound.

Another important aspect of the geometric matching problem in this domain is that many of the data features can be spurious, and thus the matches of model features with spurious data features generate spurious constraints in transformation space, adding complexity to the partition of transformations space. If it is unlikely that many spurious matches are geometrically consistent with the model then the complexity of the arrangement due to spurious matches will be low.

To exploit the inherent structure of the geometric matching problem, the conceptual and algorithmic approach we take is to decompose a transformation into its scaled-rotational component and its translational component, and explore projections of the constraints. The partition of Ω we seek is then composed of separate but inter-dependent partitions of the rotation space and the translation space. The following sections discuss this approach.

4 Decomposing Translation and Scaled-Rotation

We'll consider in some detail the case of planar 2D transformations to illustrate these ideas. We're trying to develop a method of exploring the pose equivalence classes without explicitly constructing them all. The approach we have taken is to explore two interdependent projections of the entire space: The projection of the constraints in Ω onto the t_1-t_2 plane for a fixed rotation $\mathbf{S}$, and the projection onto the s_1-s_2 plane of certain faces, edges, and vertices of the constraint hyperplane arrange-

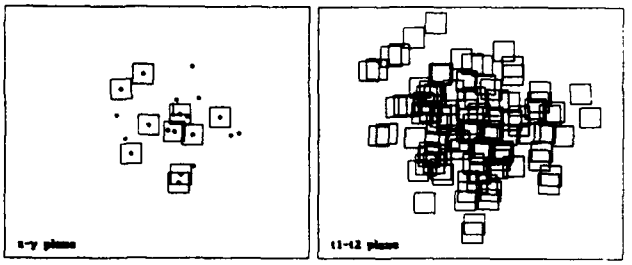


Figure 2: **Left:** Model features (open circles) after some arbitrary rotation, data features (filled circles), and uncertainty regions. **Right:** *One set of translation constraints,* C_{ij}, for each possible feature match (m_i, d_j) for the features shown on the left, for the case of the particular rotation applied to the model as shown on the left. Each square, C_{ij}, on the right is the conjunction of constraints for a single feature match representing those translation feasible for that match and the particular rotation.

ment. The algorithm we develop is based on finding maximal match sets, rather than explicitly constructing transformation equivalence classes. The following provides a brief overview of the approach.

Suppose we fix the rotational part, S, of the transformation. For fixed S the uncertainty constraints impose constraints on feasible *translations*. This imposes an arrangement of constraints in the t_1-t_2 plane as shown in figure 2. Each cell of this 2D arrangement defines a maximal match set according to the match constraints consistent with the translations in the cell. So for fixed rotation S we can read off all the maximal match sets feasible at S.

Now, as we vary S the constraints in the t_1-t_2 plane change, and the set of feasible maximal match sets changes. Intuitively, if we vary over all rotations S we can collect all feasible maximal match sets. As we'll show however, there are equivalence classes of *rotation* S and so we only have to vary S over a set of representatives of these equivalence classes. The idea is that the projection of the transformation constraints onto the s_1-s_2 plane partitions the *rotation space* into equivalence classes. Each rotational equivalence class is associated with a set of topologically equivalent projections of the constraints onto the t_1-t_2 plane (for fixed S). We'll next look at this in more detail.

4.1 Translational Equivalence Classes

Consider a specific match in the 2D case between a model point $\mathbf{p}_{m_i}$ and a data point $\mathbf{p}_{d_j}$. A transformation $T[\mathbf{p}_{m_i}] = \mathbf{S}\mathbf{p}_{m_i} + \mathbf{t}$ on the model feature is *feasible* if $\mathbf{S}\mathbf{p}_{m_i} + \mathbf{t} \in U_j^p$. Define $C_{ij}(\mathbf{S}) = \{\mathbf{t} | \mathbf{S}\mathbf{p}_{m_i} + \mathbf{t} \in U_j^p\}$ to be the set of *feasible translations* for (m_i, d_j) for any given rotation S. It is easy to see that $C_{ij}(\mathbf{S})$ is equivalent to the region U_j^p translated by $(-\mathbf{S}\mathbf{p}_{m_i})$. For a fixed rotation S, define the function $\varphi_S(\mathbf{t}) = \{(m_i, d_j) | \mathbf{S}\mathbf{p}_{m_i} + \mathbf{t} \in U_j^p\}$. Just as before, this function partitions the space of translations into equivalence classes E_k^t by $\mathbf{t} \equiv \mathbf{t}' \iff \varphi_S(\mathbf{t}) = \varphi_S(\mathbf{t}')$.

The point of this decomposition of rotation and translation is that a particular rotation S determines an arrangement of constraints in the t_1-t_2 plane of all constraint regions $C_{ij}(\mathbf{S})$ as shown in figure 2. The elements of this arrangement formed by the boundary curves of the sets $C_{ij}(\mathbf{S})$ define the equivalence classes E_k^t. The cells of this arrangement allow us to compute the maximal match sets directly: For each cell the matches in the maximal match set are given by the set of feasible translations $C_{ij}(\mathbf{S})$ which intersect the cell.

4.2 Topological Equivalence Classes of Rotation

The arrangement of the translation constraints in the t_1-t_2 plane is a function of rotation S. As we vary S, each time the topology of the t_1-t_2 arrangement changes, some new set of translation constraints becomes satisfied or an old set ceases to be satisfied. It is here where new maximal match sets appear or disappear. Intuitively the idea is to vary over all rotations and enumerate the set of all maximal match sets as they appear. We can do this exploration of rotation space in an orderly way by noting that there are equivalence classes of rotation within which the topology of the t_1-t_2 plane arrangement doesn't change, and thus the set of maximal match sets represented doesn't change. This means that there are equivalence classes of the rotation parameter S within which the same topological arrangement of the translation constraints C_{ij} is imposed in the t_1-t_2 plane, and the same set of maximal match sets is feasible. So two different rotational equivalence classes are associated with different maximal consistent match sets, although they may have most maximal match sets in common. Figure 3 shows the translation space arrangement for rotations in different rotational equivalence classes. We uncover all maximal match sets by exploring all rotations S, and collecting the different maximal match sets as they appear in the t_1-t_2 plane arrangement, but because there are equivalence classes of rotations, we need only consider a representative of each of these in this search.

The rotational equivalence classes are formed by projecting the constraints in the full transformation space onto the s_1-s_2 rotation plane. Recall that for each cell in the constraint hyperplane arrangement in the 4 dimensional transformation space a particular set of match constraints are satisfied. By moving across, say, a face or edge of these cells we change the set of constraints satisfied. Now consider the projection of the 4 dimensional transformation space onto the 2 dimensional rotation space, the s_1-s_2 plane. For an arbitrary fixed point, s, in the this plane consider the set of match constraints satisfied. For a single match, at the fixed rotation s, there is always some translation for which the constraints are satisfied for the match. This is consistent with the fact that the projection of the constraints for the match from the full 4D transformation space onto the s_1-s_2 plane covers the whole plane. In contrast, consider the intersection of constraints for two different matches. The projection of their mutually feasible transformation region onto the s_1-s_2 plane does not cover the entire plane: there are some scaled rotations s for which there is *some* translation satisfying the constraints for both matches, and there are some rotations for which there is *no* trans-

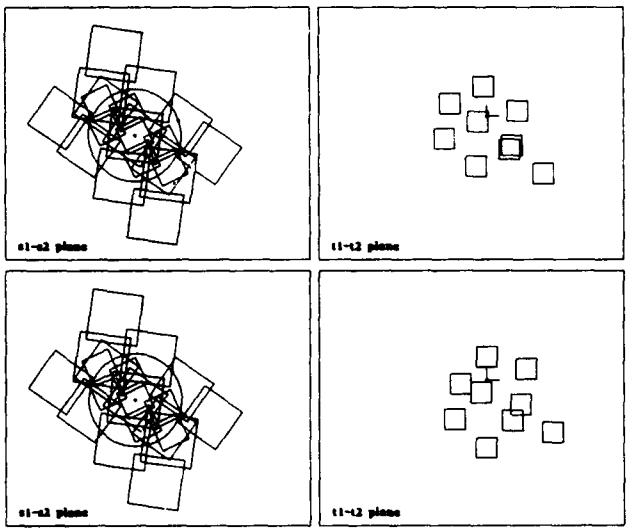


Figure 3: Two snapshots of rotation space and translation space for particular model and data sets of 3 features each, at different rotations of the model. For each pair of shots, one shows the s_1-s_2 plane position constraints and a dot on the σ-circle at the rotation applied, and the other shows associated set of t_1-t_2 plane constraints.

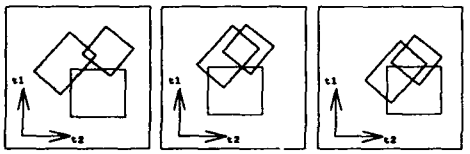


Figure 4: The translation constraints for 3 different feature matches shown for each of 3 different model rotations. For the rotation producing the left panel, there is no feasible translation for all 3 matches, while there is for the rotation producing the right panel. The middle panel is the configuration produced by rotations which lie on the intersection of 3 hyperplane constraints in Ω.

lation which will satisfy the constraints. These regions of the s_1-s_2 plane are delineated by projections of the constraints for the two matches onto the plane.

Consider the intersection of 3 constraint hyperplanes due to 2 or 3 matches (two constraints from one match and one constraint from a second match, or one constraint from each of three matches). This is a line in $\mathbf{R}^4$. Its projection to the s_1-s_2 plane is a line delineating the rotations associated with two different topological arrangements of the translation constraints for the associated feature matches. For example, if this projected line is due to the coincidence of constraints shown in figure 4 then for rotations on one side of the line it is possible to satisfy all the constraints (for some translation), and for rotations on the other it is not.

The idea is that the projection of the intersection of the constraints onto the s_1-s_2 plane partitions the rotation space into equivalence classes. Within each equivalence class the topological arrangement of constraints in the t_1-t_2 translation plane is the same. Figure 3 has panels showing the projection of constraints onto the s_1-s_2 plane for the case of point features and square uncertainty regions, (like those shown in figure 2). In this figure, each line segment in the s_1-s_2 plane is the projection of the intersection of the constraint hyperplanes for 2 feature matches. This figure is composed of several squares of different orientations arranged around the origin. the circle is centered at the origin and will be explained later. Each square is associated with a different pair of matches. The rotations inside each square are the rotations for which it is possible to align the two matches for some translation. There are several geometric events in the t_1-t_2 plane associated with the projected constraint lines in the s_1-s_2 plane. An example is shown in figure 4.

To construct the partition of the s_1-s_2 plane we simply consider all pairs and triples of feature matches, and project to the s_1-s_2 plane the points in transformation space where the constraints intersect which are associated with a boundary of their region of mutual feasibility. This forms the partition of rotation space. Points which lie in the boundary of the projected region of feasibility for pairs or triples of matches are what we are interested in most. These are points of transformation space where the satisfiability of the constraints for the pair or triple of matches changes. An approximation we make is to only project these elements of the arrangement. The others are not associated with changes in feasibility, but just the topology of the constraints.

5 Algorithms for Computing Maximal Match Sets

In this section we describe algorithms for computing maximal match sets and pose representatives forming pose hypotheses. First we consider a simple case of unoriented point features with square uncertainty regions and known scale factor σ and later show how to extend to more general cases.

The basic approach can be summarized as follows. Assume there are n data features and m model features. Given a set of model features and a set of data features and their associated uncertainty squares, form the set of all model-data feature matches. Each feature match (m_i, d_j) defines a region of feasible translations $C_{ij}(\mathbf{S})$ in the t_1-t_2 plane which is a function of the rotation $\mathbf{S}$ applied to the model feature. Each pair of feature matches and the associated regions C_{ij} and C_{kl} contribute to the partition of the s_1-s_2 plane into equivalence classes. By considering all such pairs of matches we can construct the complete partition of rotation space into equivalence classes. Each such equivalence class, or cell of the arrangement in the s_1-s_2 plane, is associated with an arrangement of the sets C_{ij} in the t_1-t_2 plane partitioning the it, and thus a particular set of maximal match sets. By passing from one rotational equivalence class to another adjacent one a local change occurs in the induced arrangement of translation constraints. We only need to analyse the part that changes, and the new maximal match sets formed, rather that re-analysing the entire t_1-t_2 plane. To construct maximal match sets we step through the cells of the the s_1-s_2 arrangement, analys-

ing the local incremental changes in the arrangement of constraints in the translation space. We simply enumerate the maximal match sets as they appear. We have two things to consider, analysis of the rotation space and the associated analysis of translation space.

5.1 Analysing the Partition of Translation Space

The geometric algorithms we develop to compute maximal match sets rely on the analysis of the two planar partitions of the scaled-rotation space and the translation space and the relationships between them. First consider the translation space. For some fixed rotation S applied to all model features, the space of translations is partitioned by the arrangement of feasible translation constraints $C_{ij}(\mathbf{S})$ for all feature matches (m_i, d_j). See figures 2 and 3. The basic component of the t_1-t_2 analysis is to compute all the elements of this arrangement in the t_1-t_2 plane formed by nm isothetic squares in the t_1-t_2 plane. The K intersecting pairs of P isothetic rectangles in the plane can be found in time $O(P \lg P + K)$. A fairly simple plane sweep algorithm using a static interval tree is described in [18]. We can extend the intersection reporting algorithm to return the set of squares C_{ij} containing each of the cells associated with each intersection point. This allows determination of the maximal match set associated with each cell. If we only report the *size* of the maximal match sets, this can be computed by the intersection reporting algorithm in $O(P \lg P + K)$[9].

When we change the rotation considered to an adjacent rotational equivalence class we have an incremental change in the constraints in the t_1-t_2 plane, which in this case involves the topological configuration between translation constraints for two matches. When we analyse this incremental change between two squares C_{ij} and C_{kl} as shown in figure 5, if there are k other squares intersecting both C_{ij} and C_{kl} we can update the change in match sets in $O(k \lg k)$ time. To facilitate these operations we require a dynamic data structure maintaining for each square C_{ij} the set of other squares it intersects.

6 Analyzing the Partition of Rotation Space

An important special case of the object localisation problem is when the scale, σ, of the object in sensory data is known. In this case the rotation consists of rotations lying on a circle of radius σ in the s_1-s_2 plane. Call this the σ-circle. The equivalence classes of rotation consist of circular arcs on the σ-circle. The *critical rotation angles* delineating the equivalence classes are those points where the σ-circle intersects the boundary curve of an equivalence class in the s_1-s_2 plane. We simply superimpose the σ-circle over the partition of rotation space constructed in the previous sections. See figure 3. Exploration of rotation space amounts to moving along the σ-circle in order of increasing angle. Each time a single line segment is crossed, exactly one topological change occurs between the regions C_{ij} in translation space[5]

[5]Note that there is a relatively rare event where more than one segment intersects the σ-circle at the same point; this requires more care to handle, but is straightforward.

An algorithm for finding representatives of equivalence classes of transformation and a measure of the size of the associated maximal match sets is as follows. Compute the positions of all feasible translation squares $C_{ij}(\mathbf{S_0})$ for some initial rotation say, $\phi = 0$. Perform an analysis of the arrangement of these squares in the translation plane as described in section 5.1. Because there are nm isothetic squares, this operation takes time $O(nm \lg mn + K)$ where K is the total number of intersecting pairs of squares. Next we perform an incremental analysis of the topological changes occurring in translation space as we step through the rotation space. There are $O(m^2n^2)$ line segments in the s_1-s_2 plane intersecting the σ-circle forming the partition of rotation space. For simplicity we assume that all intersection points of the line segments and the σ-circle are disjoint. Each intersection point is a critical rotation angle delineating a rotation equivalence class. Sort the critical rotations by angle in $O(n^2m^2 \lg mn)$ time. Step through the critical rotations in order performing the dynamic-incremental analysis of the topological changes in the t_1-t_2 plane, and record new match sets that are created. Rather than the match sets and the pose equivalence classes, we will record a representative of the transformation equivalence class and a measure of the size of the match set. As noted in section 5.1 for the i^{th} critical rotation $\mathbf{S}_i$, this incremental analysis requires time $O(k_i \lg k_i)$ where k_i is the number of squares intersecting both squares associated with the critical rotation angle $\mathbf{S}_i$. Thus the entire analysis requires time $O(n^2m^2 \lg mn) + O(nm \lg mn + K) + \sum_{h=1}^{O(m^2n^2)} O(k_h \lg k_h)$. To get a sense of this complexity, if we assume that it is unlikely that many incorrectly matched features will be simultaneously geometrically consistent, then we can approximate $k_i \sim m$. We justified this assumption by experiment as shown in section 7. Noting that $K = O(n^2m^2)$ then the above expression reduces to $O(n^2m^3 \lg m)$ for the localisation process. We can improve this bound as described in the following sections.

6.1 A Randomised Algorithm

We can use a randomised algorithm to get an expected time speedup in the algorithm by exploiting the structure of the matching problem. The idea is to explore the equivalence classes in a certain order, and in a way that allows us to make approximations that let us spend most computational effort looking only at places in transformation space where there is a high degree of geometric consistency.

Assume we have a *base match* between a model feature and a data feature, and that we compute the equivalence classes of rotation only associated with all matches paired with the base match[17]. There are $O(mn)$ other matches and so there are $O(mn)$ critical rotations. We compute this subset of the critical rotation angles, and explore this sub-partition of the rotation space as before. For each of these rotational equivalence classes we analyse the t_1-t_2 plane constraints C_{ij} which are consistent with the base match, i.e. for which C_{base} and

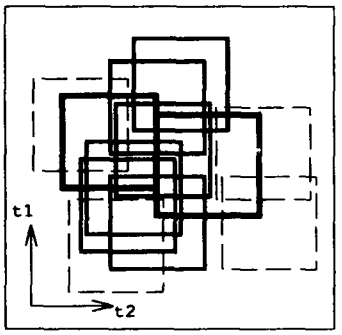


Figure 5: The incremental event in the t_1-t_2 plane to be analysed at each critical rotation due to position constraints. The critical rotation point in the s_1-s_2 plane is associated with the special alignment of constraints such that a pair of matches just becomes feasible. The bold squares are the constraints for the two matches associated with the critical rotation. New match sets are formed by the new cells created with their overlap with the k other squares affected. The dotted squares are those not involved in the analysis because no topological changes occur with them.

C_{ij} overlap. The idea is to randomly choose the base matches[12]. A simple argument shows the expected number of randomly chosen single base matches we must choose before we find a correct one is $O(n)$. Analysing the sub-partition of rotation space due to a correct base match does not mean we will find the rotation where all correct match constraints are satisfied, but it does mean that the correct matches will be *weakly consistent* with the base match, meaning there is a rotation class and critical point in the sub-partition at which each correct match constraint C_{c_i} is consistent with the base match constraint C_{base}, but where they are not necessarily all mutually consistent. However, it is very likely that we will find a rotation in this way where most if not all the correct match constraints are satisfied. Nonetheless, we are guaranteed to eventually find it, if we look through all possible base matches.

The notion of *weak consistency* as just defined can be used to define a *conservative consistency estimate*. The number of match constraints C_{ij} consistent with the base match C_{base} constraints never underestimates the size of the largest maximal match set composed of those matches. Thus this can be used to prune away parts of the transformation space where detailed translation plane analysis is unnecessary. We'll analyse in more detail the complexity of this approach in section 7.

6.2 Generalising the Algorithm

If we associate orientations with features the true orientation for data feature d_j falls within an uncertainty region U_j^θ which we will consider a range of angles $U_j^\theta = [(\hat{\theta}_{d_j} - \delta), (\hat{\theta}_{d_j} + \delta)]$ where δ is the bound on orientation uncertainty, and $\hat{\theta}_{d_j}$ is the measured orientation of d_j. We can express the angle constraints in terms of linear constraints on the transformation space: we have that $(\mathbf{S}\hat{\theta}_{m_i})^T \hat{\mathbf{n}}_j^+ \leq 0$ and $(\mathbf{S}\hat{\theta}_{m_i})^T \hat{\mathbf{n}}_j^- \leq 0$ where $\hat{\mathbf{n}}_j^+$ and $\hat{\mathbf{n}}_j^-$ are normal respectively to the two lines through the origin at angles $\hat{\theta}_{d_j} + \delta$ and $\hat{\theta}_{d_j} - \delta$ in the image plane.

Incorporating angle constraints actually complicates the algorithm described in section 6 for analysing incremental topological changes in translation space constraints. When we include angle constraints, crossing into another rotational equivalence class by crossing an angle constraint has the effect of a region C_{ij} suddenly appearing or disappearing in the t_1-t_2 plane, as the angle constraints on the associated match change state. Analysing this incremental change in arrangement requires time $O(k \lg k + k^2)$ where again k is the number of other match translation constraint regions affected by the change. In practice we get a huge speedup by including angle constraints, because this reduces the number of constraints which interact with one another, however with the simple algorithm we have outlined, considering angle constraints adds to the asymptotic complexity.

In the case of unknown scale σ we must consider the entire partition of the s_1-s_2 plane to compute all equivalence classes of transformation, not just the partition of the σ-circle. Each element of this arrangement is a rotational equivalence class and is associated with a partition of translation space as above. However because this is a projection, the complexity of analysing the s_1-s_2 plane is not asymptotically the square of the number of line segments, but less.

We note that we can easily generalise the algorithm to handle other features such as line and curve segments by defining the appropriate uncertainty regions. We can also include uncertainty ϵ and δ among the unknown parameters.

7 Experimental Implementations

There are two classes of experiments that have been conducted: experiments with real images to demonstrate the practical application of the technique, and carefully controlled experiments on synthetically generated data to investigate the computational complexity of the implemented algorithm empirically. In both cases we considered rigid planar motion (known scale), 2D model and data features consisting of 2D points and an associated unit direction vector, and isothetic square uncertainty regions. In the case of real images these were derived from points uniformly sampled on the boundary contour of the objects. In the synthetic experiments models were collections of randomly generated features and data were constructed by randomly transforming the model features, randomly perturbing their position and orientation to simulate sensor error, and adding spurious features.

The real images used were quite difficult in terms of occlusion and clutter. The algorithm was applied to about 20 different images, and successfully located the object in all but the most extremely occluded cases. Examples of these images are shown in figure 7.

The synthetic experiments were designed to determine the amount of work required to perform localisation. Recall that the basic algorithm proceeded as follows. We compute the critical rotations delineating rotation equivalence classes by considering $O(m^2n^2)$ pairs of feature matches. Each rotational equivalence class is associated

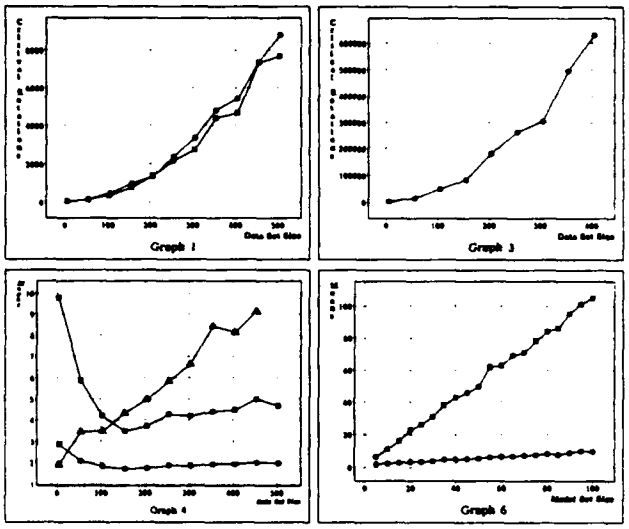


Figure 6: Graph 1: Critical rotations vs. data set size for full angle constraint (squares) and partial angle constraints (open circles). Graph 3: Critical rotations vs. data set size for no angle constraints. Graph 4: Mean values of k (circles) and k^2 (squares) vs. data set size, for the case of full angle constraints (the case of partial angle constraints is similar). Graph 6: Mean (circles) and Max (squares) values of k vs. model set size for case of full angle constraints. The cases of partial and no angle constraints are similar. The uncertainty used was $\epsilon = 8$ and $\delta = \frac{\pi}{15}$.

with a particular arrangement of translation constraints in the t_1-t_2 plane from which we can easily compute the maximal match sets valid at all rotations in the rotational equivalence class. Changing to an adjacent rotational equivalence class causes an incremental change in the constraints in the t_1-t_2 plane. This is associated with either the beginning or end of the range of feasibility of some group of maximal match sets. Figure 5 shows this event for a critical point due to position constraints. Say the incremental change in the t_1-t_2 plane due to crossing a rotational critical point affects the topological relationship of the constraints for k matches. Then if the rotational critical point was due to a position constraint then the change can be analysed in $O(k \lg k)$ time; if it was due to an angle constraint then the change can be analysed in $O(k \lg k + k^2)$ time. To understand the amount of work we have to do to perform localisation we have to know how big k is, and how many critical rotations there really are. Asymptotically we expect that $k = O(mn)$, and so *asymptotically* with the full algorithm described we would expect to do $O(m^3 n^3 \lg mn)$ work when angle constraints are not used, and $O(m^4 n^4)$ when angle constraints are used. However the asymptotic bounds describe a case in which recognition is impossible: either there is so much spurious data that the model is hallucinated all over the image, or the data are so swamped in sensor uncertainty that recognition is impossible anyway. What these experiments demonstrate is that for problem instances of practical interest the actual work done in localisation is far less than these asymptotic upper bounds.

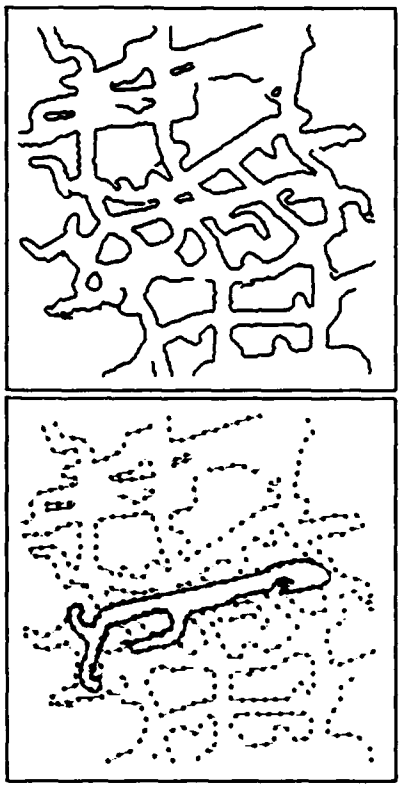


Figure 7: The edges, and the correct hypothesis, for the image shown in figure 8. The dots on the contours are the feature points used in feature matching. Images of this level of clutter and occlusion are typical.

We consider three different algorithms. One only enforces position constraints, ignoring angle constraints. Another enforces both position and angle constraints. The last uses angle and position constraints to eliminate pairs of feature matches whose constraints could never be satisfied, but uses only position constraints to partition the rotation space. Call these the case of no angle constraint, full angle constraints, and partial angle constraints, respectively.

First we'll note how many critical rotations need to be considered. Figure 6 graphs 1 and 3 show the number of critical rotations partitioning rotation space as a function of the size of the data feature set for fixed model size. As we expect there is a quadratic dependence on the data feature set size, although as can be seen by comparing graph 1 with graph 3, when angle constraints are used there is a considerable constant factor reduction in computation. Similarly for the number of critical rotations as a function of the model size for fixed data size we see quadratic dependence as expected.

Next note how much work, on average, has to be done for each critical rotation to find the size of the maximal match sets as they're uncovered. We know that at most we will have to consider the translation constraints for each of the exactly mn matches. We found empirically that for practical-size problems over a broad range of sizes the number of matches considered per critical



Figure 8: The image for the experimental results in figure 7.

rotation on average is approximately constant as both model and data set sizes vary rather than $\sim mn$, the upper bound. Figure 6, graph 4 shows the mean value of the number of match constraints in the t_1-t_2 plane affected by each critical rotation, which we called k above, for fixed model set size, as a function of the data set size. For the cases where angle constraints were used, shown in graph 4, we see that for a broad range of data set sizes of practical interest, up to 100 times the size of the model, the mean values of k and k^2 are approximately constant as the data set size increases. When angle constraints were not used, the mean value for k (for fixed model size) is approximately a very slowly growing linear function of the data set size, reasonably approximated by a constant. Figure 6 graph 6 shows these quantities for fixed data set size, as a function of the model set size. The max value of k is linear in the model size, while interestingly the mean value of k is approximately a slowly growing linear function of the model set size reasonably approximated as constant. So for problems sizes of this order we see that the assumption that $k \sim m$ is reasonable upper bound.

Let's analyse three different variations of the algorithm in more detail. For the case of partial angle constraints and no angle constraints the incremental work required to analyse the change in t_1-t_2 plane constraints associated with each critical point is $O(k \lg k)$ where again k is the number of match constraints affected as shown in figure 5. Over a large range of feature set sizes the average work per critical point is a small constant, and the amount of work required to analyse all maximal match sets is $\sim m^2n^2$. If we use the randomised algorithms we get a speedup of $\sim m$ and so the total expected work is $\sim n^2m$. Arguably, the mean value of k is a very slowly growing linear function of the data set size in the case of no angle constraints as can be seen from graph 4. If this is included in the analysis a factor of m is added to the complexity. For the case where we use full angle constraints and position constraints to partition the rotation space the incremental analysis takes up to k^2 time when the associated critical point is due to an angle constraint. From graph 4 in figure 6 we see that over a broad range of data set sizes, the mean value of k^2 is small and roughly constant as the data size increases, but was found to be quadratic in model size. So the average work per critical point is at most $\sim m^2$, and the total work is $\sim m^4n^2$ for the full algorithm or $\sim n^2m^3$ expected case for the randomised algorithm.

So we see on average the amount of work to do at each critical rotation is about constant and is not dependent on the data set size. These experiments illustrate two points. First the complexity of the constraint arrangement in high only near transformations where the model matches the data well. Elsewhere the complexity is often low. Second, because of the random nature of the spurious data, it is unlikely that (for reasonable data set size) many different large spurious feature sets will match the model, as demonstrated by the independence from n in the experiments.

This illustrates the point that the complexity of the search space is usually only high in regions near the place we're looking for, that is, a range of transformations consistent with many match constraints. If we are careful about how we explore the transformation space we can exploit this fact and utilize algorithms that have far less that the asymptotic complexity bounds, and are practical to compute.

8 Related Work

There are several recognition systems which rely on matching local geometric features to determine feasible model poses from the sensory data. We briefly mention some relevant ones. Grimson and Lozano-Pérez developed a recognition system based on searching the space of possible match sets making use of what they called an interpretation tree. In its basic form this exhaustively searches correspondence space exploiting geometric constraints to prune the search[14]. One of the reasons why this method has exponential complexity in spite of geometric constraints is that it unnecessarily searches through exponentially many subsets of what we call maximal match sets. In practice, however, by using heuristics to concentrate the search in areas where many matches are consistent the system performs well. One problem, however, is that returning a correct negative answer requires considerable search.

Baird[3] developed a method to localise 2D objects from 2D data using point features as a representation. Baird carefully formalised the problem of feature matching under uncertainty by defining uncertainty bounds for the feature locations characterised by sets of linear inequality constraints on the transformed feature positions. Important contributions of this work are the use of scaled rotations to form the linear vector space of 2D scaled rigid motions and the use of linear uncertainty constraints. Essentially his method searched an interpretation tree except it maintained global consistency using linear programming, checking the feasibility of a match set as each new match were added.

Baird reported polynomial expected case performance when there were no missing and spurious features, yet could see no way of avoiding exponential search when there were missing and spurious features. Although as

we have shown in this paper the underlying transformation space has polynomial complexity, Baird's tree search does not exploit this fact. Baird focused on a correspondence-space search, and used the matching constraints and linear programming to guide the search. He did not consider the idea that all constraints could be expressed in transformation space together delineating a polynomial number of transformation equivalence classes. Thus while his formulation permitted a polynomial-time solution, his algorithm did not exploit it.

Breuel[4] has developed an elegant correspondence-space search technique based on the problem formulation and tree search method used by Baird[3], and the notion of transformation equivalence classes and maximal match sets described in [7] and here. His algorithm exploits the constraints in transformation space imposed by each match, and the fact that there are only a polynomial number of meaningful transformation equivalence classes to arrive at a polynomial-time algorithm. The problem of exponential search is avoided by guiding the correspondence space search so that only maximal match sets are constructed. See also [17, 6, 1, 11].

9 Acknowledgments

Thanks to folks in the MIT AI Lab for helpful discussion or comments on earlier drafts of this paper. Thomas Breuel pointed out the value of Baird's[3] linear formulation of the 2D transformation and his use of linear uncertainty constraints. This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, and was funded in part by an Office of Naval Research University Research Inititative grant under contract N00014-86-K-0685, and in part by the Advanced Research Projects Agency of the Department of Defense under Army contract number DACA76-85-C-0010, and under Office of Naval Research contract N00014-85-K-0124.

References

[1] H. Alt, K. Mehlhorn, H. Wagener, and E. Welsl. Congruence, similarity, and symmetries of geometric objects. In *Discrete and Computational Geometry*, volume 3, pages 237–256. Springer-Verlag, New York, 1988.

[2] N. Ayache and O. D. Faugeras. Hyper: A new approach for the recognition and positioning of two-dimensional objecs. *IEEE Trans. PAMI*, 8(1):44–54, 1986.

[3] Henry S. Baird. *Model-Based Image Matching Using Location.* MIT Press, Cambridge, MA, 1985.

[4] Thomas M. Breuel. An efficient correspondence based algorithm for 2d and 3d model based recognition. A.I. Memo No. 1259, MIT AI Lab, October 1990.

[5] Todd A. Cass. Robust 2d model-based object recognition. A.I. Technical Report No. 1132, MIT AI Lab, 1988.

[6] Todd A. Cass. A robust implementation of 2d model-based recognition. In *Proc. IEEE Conf. CVPR*, Ann Arbor, Michigan, 1988.

[7] Todd A. Cass. Feature matching for object localization in the presence of uncertainty. A.I. Memo No. 1133, MIT AI Lab, May 1990.

[8] Todd A. Cass. Feature matching for object localization in the presence of uncertainty. In *Proc. IEE Conf. ICCV*, Osaka, Japan, 1990.

[9] Todd A. Cass. Polynomial-time object recognition in the presence of clutter, occlusion, and uncertainty. A.I. Memo No. 1302, MIT AI Lab, October 1991.

[10] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, 1987.

[11] R. E. Ellis. Uncertainty estimates for polyhedral object recognition. In *Proc. of IEEE Conference on Robotics and Automation*, pages 348–353, 1989.

[12] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[13] W. Eric L. Grimson. The combinatorics of object recognition in cluttered environments using constrained search. *Artificial Intelligence*, 44:121–166, 1990.

[14] W. Eric L. Grimson and Tomas Lozano-Perez. Localising overlapping parts by searching the interpretation tree. *IEEE Trans. PAMI*, 9(4):469–482, July 1987.

[15] Daniel P. Huttenlocher. *Three-Dimensional Recognition of Solid Objects from a Two-Dimensional Image.* PhD thesis, MIT, 1988.

[16] Daniel F. Huttenlocher and Todd A. Cass. Measuring the quality of hypotheses in model-based recognition. *To Be Published*, 1991.

[17] David W. Jacobs. Optimal matching of planar molels in 3d scenes. In *Proc. IEEE Conf. CVPR*, Maui, Hawaii, June 1991.

[18] E. P. Preparata and M. Shamos. *Computational Geometry–An Introduction.* Springer-Verlag, New York, 1985.

[19] G. Stockman, S. Kopstein, and S. Bennet. Matching images to models for registration and object detection via clustering. *IEEE Trans. PAMI*, 4(3), May 1982.

[20] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE Trans. PAMI*, 13(10):992–1007, 1991.

Data and Model-Driven Selection using Color Regions

Syeda Tanveer Fathima
Artificial Intelligence Laboratory, M.I.T Cambridge, MA 02139

Abstract

One of the major problems in model-based object recognition is selection, namely, the problem of determining which regions in the image are likely to come from a single object. In this paper we present an approach that uses color as a cue to perform selection either based solely on image-data (data-driven), or based on the knowledge of the color description of the model (model-driven). Specifically, the paper argues for the specification of color in terms of color categories, as being appropriate for the task of selection. These color categories are used to develop a fast region segmentation algorithm that extracts perceptual color regions in images. The color regions extracted form the basis for performing data and model-driven selection. Data-driven selection is achieved by selecting salient color regions. The saliency of color regions is judged by a color-saliency measure that emphasizes attributes that are also important in human color perception. The approach to model-driven selection, on the other hand, exploits the color region information in the model to locate instances of the model in a given image. The approach presented tolerates some of the problems of occlusion, pose and illumination changes that make a model instance in an image appear different from its original description. Finally, the utility of color-based data and model-driven selection is discussed in the context of reducing the search involved in recognition.

1. SELECTION IN RECOGNITION

One of the major problems in object recognition is selection, namely, the problem of identifying regions in an image within which to start the recognition process. In other words, selection is the process of isolating regions in an image that are likely to come from a single object. Model-based object recognition methods that try to recognize which of the models from their library of models are present in the scene, usually use geometric features such as points or edges and try to identify pairings between data and model features that are consistent with a rigid transformation of the object model into image coordinates. The large number of such pairings that need to be examined in cluttered scenes leads to a combinatorial search problem. It has been shown that this search can be considerably reduced if recognition systems are equipped with a selection stage where subsets of data features can be isolated that are likely to come from a single object, thus allowing search to be focused on those matches that are more likely to lead to a correct solution [4]. This isolation can be either based solely on image data (data-driven) or can incorporate the knowledge of the model (task-driven or model-driven). In addition, it is desirable to order these subsets of data features such that the more promising ones, i.e., those that are more likely to point to a single object, are explored first. This can not only increase the likelihood of a good match being obtained earlier, but is also useful when the task is to recognize as many objects as possible in a scene. Thus the goals of selection in recognition are two fold: To isolate areas in the image that are likely to come from a single object, and to order these regions such that the more promising ones are explored first. From this, it is apparent that the goals of selection are different from those of segmentation, where the problem is to partition the image into regions that contain a single object. In selection, on the other hand, it is not essential to isolate regions that totally contain a single object, nor is it necessary to partition the entire image into different object containing regions.

Even though selection can be of help in recognition, it has largely remained unsolved. What makes selection so difficult? In the ideal case, if the appearance of the desired object in the scene was known, and objects in the scene were nicely separated and distinguishable from the background, and the illumination conditions were known, then even simple methods that rely on intensity measurements would work well to extract groups of features. But in reality the appearance of the object is not known. In addition, illumination conditions and surface geometries of objects present in a scene can cause problems of occlusion, shadowing, specularities, inter-reflections, etc. in the image and make it difficult to interpret groups of data features such as edges and lines. Previous approaches to selection have focused on the problem of data-driven selection by grouping data features such as edges, lines, points, etc., based on con-

straints such as parallelism, collinearity, etc. [8], distance and orientation [7], and regions enclosed by a group of edges [1]. The extent to which such grouping methods reduce the search in recognition depends on the reliability of the groups produced (i.e. how many of them really come from a single object). With the constraints used, maintaining the reliability of groups was found to be difficult. So the general problem of selection remains largely unsolved as it is still not obvious how to reliably characterize subsets of data features that will give clues that point to a single object. Thus it appears that there is a need for a computational model of selection to explain both data and task-driven selection.

We have been involved in building one such model that proposes that selection be accomplished via an attention mechanism. Specifically, it is an attempt to build a computational model of the visual attention phenomenon in humans, and to propose it as a general purpose selection mechanism in recognition. This involves the isolation of two modes of human attentional behavior, namely attracted-attention and pay-attention modes, to serve as paradigms for data-driven and model-driven selection respectively. The *attracted-attention* mode of behavior is spontaneous and is commonly exhibited by an unbiased observer (i.e., with no *a priori* intentions) when some objects or some aspects of the scene attract his/her attention, while the *pay-attention* mode is a more deliberate behavior exhibited by an observer looking at a scene with *a priori* goals (such as the task of recognising an object, say) and hence paying attention to only those objects/aspects of a scene that are relevant to the goal. According to this model, therefore, data-driven selection can be achieved by identifying regions in an image that attract attention (i.e., that are distinctive) with respect to some feature such as color, texture, etc., while model-driven selection can be achieved by paying attention to the model features (i.e., using the model features to decide saliency of features in the image). While it is understandable that paying attention to model features can help isolate areas in the image that could contain subsets of data features that are likely to contain a single object (or the specific model object in this case), it is not immediately apparent how locating salient regions can help in serving the goals of selection. Such a choice is, however, motivated by the following considerations. First, it is often observed that an object stands out in a scene because of some distinctive features that are usually localised to some portion of the object. Therefore isolating distinctive regions is more likely to point to a single object. Secondly, a distinctive region, if suitably found, can help in limiting the number of candidate models from the library that can potentially match the given data. This is especially true if only a few models in the library satisfy the features that made the data region distinctive. Lastly, it has often been observed that the first objects recognized in a scene are those that attract an observer's attention. Thus ordering the regions by distinctiveness to decide which objects to recognize first seems to be in keeping with this observation. Finally, a number of other approaches have also suggested that selection, at least data-driven, can be performed based on some measure of saliency, such as the structural saliency of curves [16], saliency defined by local differences in contrast, color, size, etc. [2],[3],[15].

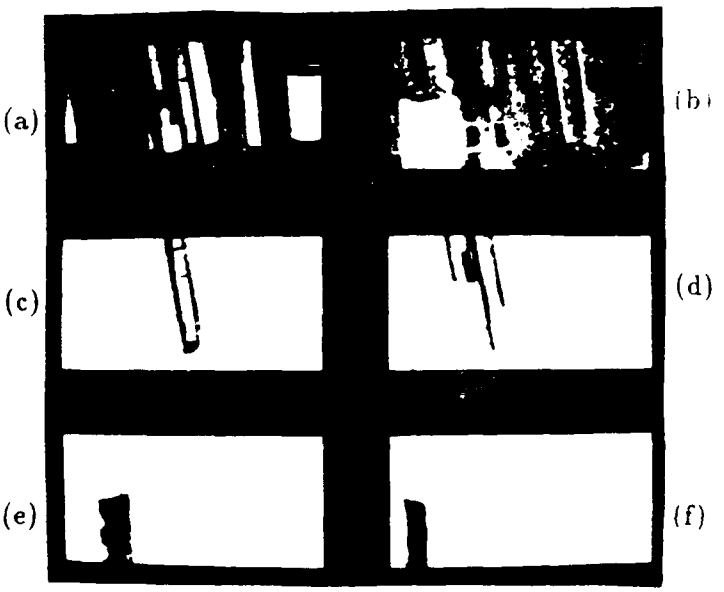


Figure 1: *Illustration of color region segmentation and color-saliency. (a) Input image depicting a scene of objects of different materials and having occlusions and inter-reflections. (b) Segmented image using the color region segmentation algorithm. (c)-(f) The four most distinctive regions detected using the color-saliency measure. The white portion in the red book appears so because of the white background.*

The above discussion indicates a framework in which data and model-driven selection can be achieved. But how can salient regions be found in the image independent of the model, and how can the object model affect the choice of regions? The purpose of this paper is to present a method of selection by restricting attention to one particular feature, namely, color. It shows how color regions can be extracted from the image and how they can be used to perform data-driven and model-driven selection. To give a flavor for the ensuing discussions, Figures 1-2 show some examples of the results of data and model-driven selection performed by our system. Figure 1a shows an image of a realistic indoor scene with shadows, inter-reflections, and consisting of many types of objects. The different color regions found in this image are re-colored and shown in Figure 1b. The four most salient color regions found are shown in Figures 1c-1f. These regions span objects in the scene that are salient in color. Figure 2 shows model-driven selection using color, using the model object shown in Figure 2a and the scene depicted in Figure 2c. The cluster of regions found to best satisfy the model color region description using our algorithm for model-driven selection is shown in Figure 2f.

The rest of the paper discusses how this kind of se-

lection can be achieved using color. It is organized as follows. In Section 2, we motivate the choice of color as a feature to study selection, and outline the requirements imposed by selection on any method for the extraction of color information. Based on these guidelines, an approach to extracting color regions is presented in Section 3. In section 4, a measure for expressing the saliency of color regions is presented and its effectiveness for data-driven selection is examined. Section 5 presents a way to perform model-driven selection based on the color regions. Finally, Section 6 summarises our approach to color-based selection.

2. COLOR IN SELECTION

2.1 Role of Color in Selection

Color is known to be a strong cue in attracting an observer's attention. Humans often also use color information to search for specific objects in a scene. It therefore seems natural to use color as a cue for performing selection in computer vision. But the strong motivation for using color in selection comes from the fact that it provides region information and that, when specified appropriately, it can be relatively insensitive to variations in normal illumination conditions and appearances of objects. A color region in the image almost always comes entirely from a single object, giving, therefore, more reliable groups than existing grouping methods and this can be useful for data-driven selection. Because objects tend to show color constancy under most illumination conditions, color can be a stable cue for most poses (appearances) of objects in scenes, thus making it also suitable for model-driven selection.

2.2 Surface Color, Image Color, Perceptual Color

Although color is useful for selection, the problem of specifying the perceived color of objects from an image of the scene has proven to be difficult. Several artifacts such as specularities (from shiny surfaces in the scene), interreflections, shading on surfaces, and shadowing all make it difficult to recover the actual color of objects in the scene from the image. Existing approaches have mainly focused on the problem of color constancy, where the goal was to extract surface color, i.e., surface reflectance properties of objects, in order to obtain a stable perception of the color of an object under varying illumination conditions. As this problem is under-constrained, most methods make some assumptions about either the surface being imaged [12], or about the illumination conditions [13],[9]. Other approaches also exist that try to recover *image color*, i.e., the color of the objects as they appear under the present illumination conditions, accounting separately for artifacts such as specularities on shiny surfaces [11]. These methods, however, cannot ensure that the color extracted matches the perceived color of regions.

For the purposes of selection, what kind of color information should be extracted from regions? Is recovering image color sufficient or should one attempt to recover surface color? We propose that for both data and model-driven selection, it is sufficient if a region could be specified by its perceived color, and the effects of artifacts such as specularities could be separately accounted as was done by image color recovery methods. Using the perceptual color, two adjacent color regions would be distinguished if their perceived colors were different, and this is sufficient for data-driven selection. Because objects tend to obey color constancy under most changes in illumination, their perceived color remains remains more or less the same thus making it sufficient also for model-driven selection. But can perceptual color be quantified at all? In general, several effects such as simultaneous color contrast, color filling, etc. have been known to influence human perception of color. Fortunately, (as we will explain later,) these factors are not very critical for selection.

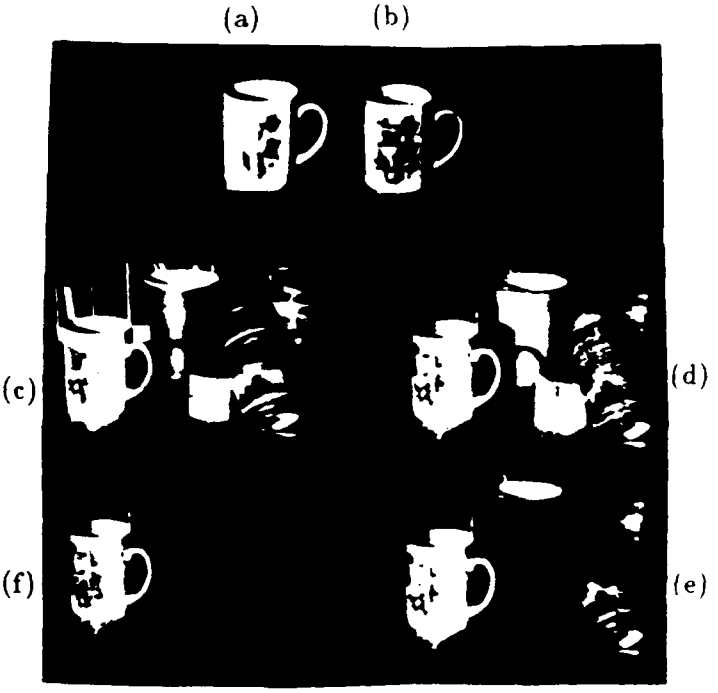


Figure 2: *Illustration of color-based model-driven selection. (a) The object serving as the model. (b) Its color description produced by the segmentation algorithm of Section 3. (c) A cluttered scene in which the object appears. (d) Regions selected based on unary color constraint. (e) Regions of (d) pruned after using the unary size constraint. (f) Regions corresponding to the best subgraph that matched the model specifications.*

2.3 Perceptual Color Specification by Categories

In this section we present a method of specifying the perceptual color of image regions from the colors of their constituent pixels. The color of pixels in images is described by a triplet <R,G,B> (called *specific color* henceforth), representing the components of image intensity at that point along three wavelengths (usually red, green and blue as dominant wavelengths to correspond to the filters used in the color cameras). When all possible triples are mapped into a 3-dimensional color space with axes standing for the pure red, green and blue respectively, we get a color space that represents the entire

spectrum of computer recordable colors. Such a color space, must therefore, be partitionable into subspaces where the color remains perceptually the same, and is distinctly different from that of neighboring subspaces. Such subspaces can be called *perceptual color categories*. Now each pixel in the image maps to a point in this color space, and hence will fall into one of these categories. *The perceptual color of this pixel can, therefore, be specified by this color category.* To get the perceived colors of regions from the perceptual color of their constituent pixels, we observe the following. Although the individual pixels of an image color region may show considerable variation in their specific colors, the overall color of the region is fairly well-determined by the color of the majority of pixels (called *dominant color* henceforth). *Therefore, the perceived color of a region can be specified by the color category corresponding to the dominant color in the region.*

The category-based specification of perceptual color (of pixels or regions) is a good compromise between choosing the specific color (which is extremely unstable w.r.t to changes in illumination conditions, etc.) and surface color (whose recovery is hard). Since the categories indicate the perceptual color, they have the same beneficial effect as recovering perceptual color, on both data and model-driven selection, such as giving a reliable segmentation of image into color regions, being stable under changes in illumination conditions, etc. In addition, since the perceptual categories depend on the color space and are independent of the image, they can be found in advance and stored in, say, a look-up table. Finally, a category-based description is in keeping with the idea of perceptual categorization that has been explored extensively through psychophysical studies. These studies concluded that although humans can discriminate between several thousand nuances of colors, psychophysically, we seem to partition the color space into relatively few distinct qualitative color sensations or categories [17].

2.4 Categorization of Color Space

The above discussion argued for the viability of an approach that recovers a color to within a category. Before this can be turned into a computational method of color recovery one needs to address the issue of how such categories may be found. Previous work on color categorization involved experiments of naming the color using a limited vocabulary, or identifying colors using the Munsell color charts [20]. But for computational color recovery, we need a way to convert the camera recordable red, green and blue components of colors into computer recordable perceptual color categories. This was done by performing some rather informal but extensive psychophysical experiments that systematically examined a color space and recorded the places where qualitative color changes occur, thus determining the number of distinct color categories that can be perceived. For this, the hue-saturation-value color space was used as it specifies a given color in terms of its hue, purity and brilliance - attributes that have been found to give a perceptual description of color [10]. The details of these experiments will be skipped here, except to mention the following. The entire spectrum of computer recordable colors (2^{24} colors) was quantized into 7200 bins corresponding to a 5 degree resolution in hue, and 10 levels of quantization of saturation and intensity values. The color in each such bin was then observed by displaying a mondrian (a uniform color patch) of that color on a monitor screen and observing it under dark room conditions with appropriate monitor calibration. From our studies, we found about 220 different color categories were sufficient to describe the color space. The color category information was then summarized in a *color-look-up table*. Although it is true that a finer level of quantization would have yielded more categories, a smaller set is actually more useful since it gives a reasonably coarse description of the color of a region thus allowing it to remain the same for some variations in imaging conditions. In fact, by the above method we can also determine which categories can be grouped to give an even rougher description of a particular hue. This was done and stored in a *category-look-up table* to be indexed using the color categories given by the color-look-up table.

3. COLOR REGION SEGMENTATION

The previous section described how to specify the color of regions, when once they have been isolated. But the more crucial problem is to identify these regions. In this section, we show that the perceptual categorization principle can be used to determine which pixels can be grouped to form regions in an image. If each surface in the scene were a mondrian, then all its pixels would belong to single color category, so that by grouping spatially close pixels belonging to a category, the desired segmentation of the image can be obtained. But real surfaces being hardly mondrians, it is rare that pixels of a region from such surfaces, all belong to the same color category. They could show considerable variation in color with bright and dark pixels intermixed, and with possibly spurious pixels also being present. We now analyse some of the color variations across an image that can result from imaging a colored surface in the scene.

3.1 Variation of Color Across an Image of 3D-Surface

In this section we use some assumptions to show that the color variations across an image of surface is mostly in intensity. When a surface is imaged, the light falling on the image plane (image irradiance) is related to the physical properties of the scene being imaged via the image irradiance equation given below:

$$I(\lambda, r) = \rho(\lambda, \mathbf{r}) F(\mathbf{k}, \mathbf{n}, \mathbf{s}) E(\lambda, \mathbf{r}) \qquad (1)$$

where λ is the wavelength, $\mathbf{r}$ is the spatial coordinate and r is its projection in the image, $E(\lambda, \mathbf{r})$ is the intensity of the ambient illumination, $\rho(\lambda, \mathbf{r})$ is the component of surface reflectance that depends only on the material properties of the surface (and hence specifies its surface color), while $F(\mathbf{k}, \mathbf{n}, \mathbf{s})$ is the component of surface reflectivity that depends on surface geometry, with $\mathbf{k}$,$\mathbf{s}$,$\mathbf{n}$ being the viewer direction, the source direction and the surface normal respectively. Although the image irradiance equation assumes that all surfaces in a scene reflect

light governed by a single reflectivity function, we can easily reinterpret this equation to represent image irradiance of a single surface. Under the assumption of a single light source, the surface illumination $E(\lambda, \mathbf{r})$ can be separated as a product of two terms $E_1(\lambda)$ and $E_2(\mathbf{r})$, and since $F(\mathbf{k}, \mathbf{n}, \mathbf{s})$ is a function of position $\mathbf{r} = F(\mathbf{r})$ the image irradiance equation can be re-written as

$$I(\lambda, r) = \rho(\lambda, \mathbf{r}) F(\mathbf{r}) E_1(\lambda) E_2(\mathbf{r}) \quad (2)$$

The surface reflectance and hence the resulting appearance of a surface is determined by the composition as well as the concentration of the pigments of the material constituting the surface. For most surfaces, the composition of the pigments can be considered independent of their concentration so that the spectral reflectance $\rho(\lambda, \mathbf{r})$ can be written as a product of two terms $\rho_1(\lambda)$ and $\rho_2(\mathbf{r})$. Note that this assumption is less restricting than the assumption of homogeneity that has been used before [9]. With this simplification, (and grouping the product of terms dependent on λ and $\mathbf{r}$ separately) the image irradiance equation becomes

$$I(\lambda, r) = H(\mathbf{r}) L(\lambda) \quad (3)$$

Now, if we consider the filtered version of this signal, i.e., the image irradiance in three channels, say the red, green and blue channels with their associate transfer functions $h_R(\lambda), h_G(\lambda), h_B(\lambda)$, the specific color at each pixel location r is specified by the triple $<R(\mathbf{r}), G(\mathbf{r}), B(\mathbf{r})>$ where

$$R(r) = \int_0^\infty I(\lambda, r) h_R(\lambda) d\lambda = H(\mathbf{r}) R_1 \quad (4)$$

$$G(r) = \int_0^\infty I(\lambda, r) h_G(\lambda) d\lambda = H(\mathbf{r}) G_1 \quad (5)$$

$$B(r) = \int_0^\infty I(\lambda, r) h_B(\lambda) d\lambda = H(\mathbf{r}) B_1 \quad (6)$$

This shows that under the given assumptions (which include non-homogeneous surfaces,) the color of a surface can vary only in intensity. In practice, even when the separability assumption on reflectance is not satisfied, or there is more than one light source in the scene, the general observation is that the intensity and purity of colors get affected but the hue still remains fairly the same. In terms of categories, this means that different pixels in a surface belong to *compatible categories*, i.e. have the same overall hue but vary in intensity and saturation. Conversely, if we group pixels belonging to a single category, then each physical surface is spanned by multiple overlapping regions belonging to such compatible color categories. These were the categories that were grouped in the category-look-up-table mentioned in Section 2.4. The next section describes how these concepts can be put together to give a color image segmentation algorithm.

3.2 Color Region Segmentation Algorithm

The algorithm for color image segmentation performs the following steps.(1) First, it maps all pixels to their categories in color space. (2) It then groups pixels belonging to the same category, (3) and finally merges overlapping regions in the image that are of compatible color categories.

1. Mapping pixels to categories: This is done by a simple indexing of the color-look-up-table by the color of the pixel specified in terms of its hue, saturation, and brightness components. These components can be derived from the specific color as described in [21]. This step takes time = O(N) where N is the size of the image.

2. Grouping pixels of same category: The image is divided into small non-overlapping bins of fixed size (say, 8x8) and the color categories found in the bins are recorded. Then a sequential non-recursive labeling algorithm is used to simultaneously assemble all the overlapping connected components using the category description in the bins. This algorithm is an extended version of the labeling algorithm for binary images earlier described[5], and uses the union-find data structure to efficiently merge category labels into connected components taking time = $O(\mathbf{k}^2\ M)$ where M = window-size, and k = maximum number of categories present in the window (= O(1) for small window-sizes, eg.,. M = 64). The resulting labels are propagated back to the pixels to give the precise boundaries of color regions of single color categories. The color of the region is then specified by the color category and specific color of the dominant color in the regions as described in Section 2.3.

3. Merging overlapping regions: The general problem of determining which regions overlap in image can be a computationally intensive operation as it involves determining which polygonal regions intersect and finding their regions of intersection. But by using the bin-wise representation of connected components, we can detect and combine overlapping regions with greater ease. From the discussion in Section 3.1, a shaded region maps to categories in color space that are compatible, i.e., have the same overall hue. The categories that are compatible are available from the category look-up-table described in Section 2.4. To find all such regions that have compatible categories and overlap in image space, the algorithm examines each window of the image to see if it contains the interior portions of regions of compatible color categories. Such overlap regions are grouped as in step 2. This step again takes $O(\mathbf{k}^2 M)$ time. Finally, the window-level color labels are propagated back to the corresponding pixels to give an accurate localization of the color region boundaries.

The algorithm for color image segmentation thus makes only a constant number of passes through the image, each being linear in the size of the image.

3.4 Results

Figures 3-4 demonstrate the color region segmentation algorithm. Figure 3a shows a 256 x 256 pixel size image of a color pattern on a plastic bag. The folding on the bag and its plastic material together give a glossy appearance in the image as can be seen in the big S and Y. The result of step-2 of the algorithm is shown in Figure 3b, and there it can be seen that the glossy portions on big blue Y and the red S cause overlapping color regions. These are merged in step 3 and the result is shown in Figure 3c. As can be seen in the figure, the algorithm achieves a fairly good segmentation of the scene for such surfaces. Figure 4a shows another image consisting of colored pieces of cloth with the textured regions having several small colored regions within them. The results of

the algorithm (Figure 4b) show that even such colored regions can be reliably isolated. Another example (Figure 1) of color region extraction was mentioned earlier in Section 1. Notice in the segmented image of Figure 1b that adjacent objects of same perceptual color are merged (grey books). This is to be expected because the grouping of regions is based on color information alone.

4. COLOR-BASED DATA-DRIVEN SELECTION

The segmentation algorithm described above gives a large number of color regions. Some of these may span more than one object, while some come from the scene clutter (background, etc.) rather then objects of interest in the scene. It would be useful for the purposes of recognition, to order and consider only some of these regions so that by isolating data subsets from such regions, the search can be focused on key groups of features thus excluding much of the scene clutter. Based on the rational given in Section 1, we propose that the color regions be ordered by their saliency, i.e., by how distinctive they appear. The method of color-based selection, therefore, is to extract color regions from the image, order them based on a measure of color-saliency and then select a few most salient regions to be given to any recognition system. In this section we first describe a measure of expressing color saliency, and then examine the utility of salient-region selection in recognition.

4.1 Finding Salient Color Regions in Images

In trying to express distinctiveness, one encounters the question: Is distinctiveness expressible at all? In general, any judgement of distinctiveness has both a sensory and a subjective component. Thus for example, while most of us can perceive brighter colors more easily than duller colors, the judgement of which of two hues of the same brightness and saturation are more salient can be subjective. The aim here is to focus on the sensory component of distinctiveness and hence extract properties of regions that are general enough to be perceived by most observers. Accordingly, we propose that the saliency of a color region be composed of two components, namely, *self-saliency* and *relative saliency*. Self-saliency determines how conspicuous a region is on its own and measures some intrinsic properties of the region, while relative saliency measures how distinctive the region appears when there are regions of competing distinctiveness in the neighborhood.

In order to develop such a measure for color-region saliency one has to ask the following questions: What features in regions determine their saliency? How can they be measured to reflect our sensory judgments? an finally, how can they be combined to give the saliency measure? We now address these questions and derive a measure of color-saliency.

4.1.1 Features used for measuring saliency

Since the saliency of a color region depends on the region features used, they must be carefully selected. Such features should be: (i) perceptually important, (ii) easily measurable, and (iii) fairly general to avoid subjective bias.

1. Color: The color of a region is an intrinsic property and affects a region's self-saliency. It is specified by (s(R),v(R)), where s(R) = saturation or purity of the color of region R, and v(R) = brightness, and $0 \leq$ s(R),v(R) ≤ 1.0. The hue of colors is not considered to avoid subjective bias.

2. Region size: The size of a region is again an intrinsic property and affects its self-saliency. It is chosen as feature based on the observation that regions that are either very small in extent, or that are large enough to cover the entire field of view, do not often attract our attention. Also, very large regions can potentially span more than one object, making them unsuitable for selection. The size feature is expressed by the normalized size r(R) = Size(R)/Image-size.

3. Color contrast: The color contrast a region shows with its neighbors, affects its relative-saliency. The rationale behind choosing color contrast is that even if a region has an interesting intrinsic color it may not be distinctive if all its neighbors also have equally interesting colors, unless it showed the greatest contrast. It is difficult to express color contrast in a numerical measure that can account for the variations in an observer's judgement with the conditions of observation, size, shape, and absolute color of the stimuli. Among the empirical formulas designed to predict the observed color differences, the cie-distance d(C_R,C_T) between two color region R and T with specific colors as $C_R = (r_0, g_0, b_0)^T$ and $C_T = (r, g, b)^T$ given by $d(C_R, C_T) = \sqrt{(\frac{r_0}{r_0+g_0+b_0} - \frac{r}{r+g+b})^2 + (\frac{g_0}{r_0+g_0+b_0} - \frac{g}{r+g+b})^2}$ has gained acceptance. But even this measure does not the take into account the hues of the colors explicitly. In the color contrast measure we chose, it is first ascertained whether the hues of the two regions are different, and then the extent of difference is judged used the cie-distance measure $d(C_R, C_T)$ in such a way that the contrast between regions of different hue is emphasized. The measure is given by c(R,T) below:

$$c(R,T) = \begin{cases} k_1 d(C_R, C_T) & \text{if R and T are of same hue} \\ k_2 + k_1 d(R,T) & \text{otherwise} \end{cases} \quad (7)$$

where $k_1 = \frac{0.5}{\sqrt{2}}$ and $k_2 = 0.5$, so that $0 \leq$ c(R,T) ≤ 1.0.

4. Size contrast: The size contrast is a feature for determining relative saliency and is chosen because it determines if a region is mostly in the background or in the foreground. The size contrast of a region R with respect to an adjacent region T is simply the relative size (area) and is given by $t(R,T) = min\left(\frac{\text{size(R)}}{\text{size(T)}}, \frac{\text{size(T)}}{\text{size(R)}}\right)$.

Since a region R has several neighboring regions in general, the color contrast $c(R)$ and size contrast $t(R)$ of a region R are measured relative to a *best* neighbor T_{best} for each region, so that $c(R) = c(R, T_{best})$, and $t(R) = t(R, T_{best})$. T_{best} is the neighboring region that is ranked the highest when all neighbors are sorted first by size, then by extent of surround, and finally by contrast (size or color contrast as the case may be).

4.1.2 Combining features for self-saliency: To de-

termine self-saliency from the chosen features, they are weighted appropriately to reflect their importance. The self-saliency measure chosen emphasizes purer and brighter colors over darker and duller colors by choosing the weighting functions for saturation and brightness as $f_1(s(R)) = 0.5s(R)$, and $f_2(v(R)) = 0.5v(R)$ respectively. The size of a region is given a non-linear weight to deemphasize-emphasize both very small and very large regions as they do not often attract our attention. The corresponding weighting function has sharp as well as and smoothly rising and falling phases determined by the breakpoints t_1, t_2, t_3, t_4 as given in the equation below.[1] Here n stands for the region size r(R).

$$f_3(n) = \begin{cases} -\frac{ln(1-n)}{c_1} & 0 \le n \le t_1 \\ 1 - e^{-c_2 n} & t_1 < n \le t_2 \\ s_2 - c_3 ln(1 - n + t_2) & t_2 < n \le t_3 \\ s_3 e^{-c_4(n-t_3)} & t_3 < n \le t_4 \\ 0 & t_4 < n \le 1.0 \end{cases} \quad (8)$$

where $t_1 = 0.1$, $t_2 = 0.4$, $t_3 = 0.5$, $t_4 = 0.75$, $s_1 = 0.8$, $s_2 = 1.0$, $s_3 = 0.7$, $s_4 = 10^{-3}$ and $c_1 = -\frac{ln(1-t_1)}{s_1}$, $c_2 = -\frac{ln(1-s_1)}{t_1}$, $c_3 = -\frac{(s_2-s_3)}{ln(1+t_2-t_3)}$, $c_4 = -\frac{ln\frac{s_4}{s_3}}{(t_4-t_3)}$ and n = size of region R = r(R).

4.1.3 Combining features for relative saliency:

Once again, the chosen features are weighted appropriately to determine relative saliency. The color contrast is weighted linearly by a function $f_4(c(R)) = c(R)$, to emphasize regions showing greater contrast. The relative size is exponentially weighted by a function $f_5(t(R)) = 1 - e^{-12t(R)}$ to favor situations in which a region and its best neighbor have approximately the same size.[2]

4.1.4 Finding self and relative saliency

Once the various features determining self and relative saliency are appropriately weighted, they reinforce each other so that the self and relative saliencies can be given by simple additive combinations of their individual features. The self-saliency of a region R denoted by SS(R) is given as $f_1(s(R)) + f_2(v(R)) + f_3(r(R))$. Similarly, the relative saliency of the region R, RS(R) is given by $f_4(c(R)) + f_5(t(R))$. Finally, the overall-saliency of a region R is expressed by a linear combination of self and relative saliency as SS(R) + RS(R), using the following rationale. Any combination method should be flexible enough to allow a region to be declared salient if it shows good contrast (i.e. high relative saliency) even though it may not be interesting on its own. Conversely, a region that is interesting on its own but fails to become interesting in the presence of neighboring regions should not be chosen. On the basis of these observations alone, nonlinear combining methods such as $(SS(R) * RS(R))$ or $\max(SS(R), RS(R))$ are not suitable. If a region is both interesting on its own as well as in the presence of other regions in the scene, then it must be given more importance. All three criteria are satisfied when the two saliency components are linearly combined. The color saliency of a region R is therefore given by

$$\text{Color-saliency}(R) = f_1(s(R)) + f_2(v(R)) + f_3(r(R)) + f_4(c(R)) + f_5(t(R)) \quad (9)$$

The saliency measure described above does not take into account perceptual effects such as simultaneous color contrast, color-filling, etc. Because such effects do not greatly undermine a region that is already very outstanding (very salient), and because saliency is being used to rank the regions, we have ignored these effects.

The color regions in the image can now be ordered using the saliency measure and a few most significant regions can be retained for selection (called salient region, henceforth). The number of salient regions to be retained can be determined when the selection mechanism is integrated with a recognition system to perform a specific task, and is therefore, left unspecified here.

4.1.5 Results

We now illustrate the ranking of regions produced by the color saliency measure derived above. Figure 1c-1f show the four most distinctive regions found by applying the color-saliency measure to all the color regions extracted from the scene shown in Figure 1a. Figures 3d-3f, 4c-4f, show the few most salient regions found in their respective scenes. In the experiments done so far, the color-saliency measure was found to select fairly large bright-colored regions that showed good contrast with their neighbors, and appeared perceptually significant.

4.2 Use of Salient Color-based Selection in Recognition

Data-driven selection based on salient color regions is primarily useful when the object of interest has at least one of its regions appearing salient in the given scene. In such cases, the search for data features that match model features can be restricted to the salient regions, thus avoiding needless search in other areas of the image. By selecting salient color regions, we get a small number of groups (a region is itself a group), containing several features. It was shown in [22] that such large-sized groups are useful for indexing, i.e. to determine which regions from models in a library could correspond to a given group. But when the task is to recognize a single object, it is desirable to have small-sized groups. For this, existing grouping techniques can be applied to the data features found within the color regions to obtain reliable small-sized groups.

We now estimate the search reduction that can be achieved with such a selection mechanism. Let (M,N) = total number of features (such as edges, lines, etc.) in the model and image respectively. Let (M_R, N_R) = total number of color regions in the model and image respectively. Let N_S = number of salient regions that are retained in an image. Let g = average size of a group of data features, within a model or image. Let $(G_M, G_N$

[1] Such a function along with the thresholds and rates of change was empirically derived from informal psychophysical experiments (whose details will be skipped here) performed using color regions of various sizes.

[2] Once again this function was obtained by performing informal psychophysical experiments.

= number of groups formed (using any existing grouping scheme) in the model and image respectively. Finally, let G_{N_i} be the number of groups in the salient image region i. Using the alignment method of recognition [6], at least three corresponding data features are needed to solve for the pose (appearance) of the model in the image. If no selection of the data features is done, then the brute-force search required to try all possible triples is $O(M^3N^3)$. If selection is done by only grouping methods (i.e., without color region selection), then the number of matches that need to be tried is $O(G_M G_N g^3 g^3)$ since only triples within groups need to be tried. But as we mentioned before, grouping methods often make mistakes, so that not all groups contain features belonging to a single object. In at least one such study [1] out of the 150 or so groups isolated, about 83 groups actually came from single objects. Most of the remaining 67 groups would not yield any consistent match and would represent fruitless search. Consider the case when grouping of data features is done within all the color regions. With this, the grouping is more reliable, and also, the number of groups is smaller (as groups straddling regions are not considered), so that the overall effect is to reduce the search. For example, with M = 200, N = 3000, g = 7, and $G_M = 30$, $G_N = 430$ (these numbers are typical of indoor scenes), the search reduction assuming 70% reliability in simple grouping to > 95% reliability in grouping within color regions is $\approx 0.25 * 10^9$ which is a considerable improvement. Consider next, when grouping is restricted to salient color regions. The number of matches further reduces to $O(\sum_{j=1}^{N_s} G_{N_j} G_M g^3 g^3)$, since only the groups in the salient regions need be tried.

To get an estimate of the number of matches and time taken for matching in real scene when color-based selection is used, we recorded the number of regions (obtained by applying the segmentation algorithm of Section3), and the number of data features within regions in some selected models and scenes (Figure 1 and 2 show typical examples of models and scenes tried). The regions were ordered using the color saliency measure and the four most salient regions were retained. Then search estimates were obtained using the above formulas, and assuming a grouping scheme that gives a number of groups within regions that is bounded by $\frac{\text{the number of features in a region}}{\text{average size of the groups in a region}}$. This is a good bound on the number of groups produced using simple grouping schemes such as grouping 'g' closely-spaced parallel lines in the region. The result of such studies is shown in Table I. As can be seen from this table, the number of matches is always smaller when salient color regions are used for selection. But the ultimate utility of such a selection mechanism can be accurately guaged only after it is integrated with a recognition system. Current research is being directed towards this effort.

5. COLOR-BASED MODEL-DRIVEN SELECTION

The previous section described a data-driven selection mechanism that was meant for an object of interest having some salient color regions. This will not be of much help when the object of interest is not salient in color (but salient in some other domain, say texture) or is not salient at all. In such cases, the color description of the model can be used to perform selection. We now describe one such color-based model-driven selection mechanism. Here, given a color-based description of a model object, the task is to search and locate color regions that satisfy this description. The use of model information to constrain the matching of model features to image features is not new. Several model-driven search restriction techniques such as generalized Hough transforms, heuristic termination, focal features, etc. have evolved [4]. The emphasis in these methods was on geometric constraints that can prune the search space during the matching stage of recognition. The approach we present here, on the other hand, emphasizes some global relational information about model color regions to prune the search space prior to matching. It also provides possible correspondences between model and image regions. Such a correspondence can further reduce the complexity of recognition because the search for pairing model features to data features can be restricted now to these corresponding regions rather than all image regions. Color information in the model object has been used before to search for instances of the object in the given image of a scene [19],[18]. These approaches represent model and image color information by color histograms and perform a match of the histograms. Such approaches usually cause a lot of false positive identifications, and do not explicitly address some of the problems that arise in going from a model object to its instance in a scene. Also, since they do not supply correspondence between model and image regions, they are not as useful for reducing the search in recognition.

In order for any scheme for model-driven selection to be effective for reducing the search in recognition, it must meet two requirements: (i) it must be sufficiently selective to avoid lot of false positive identifications that cause needless search for matches. (ii) it must be sufficiently conservative to avoid lot of false negatives, causing recognition to fail when it should have succeeded. A selection scheme can make false negatives if it does not adequately take into account the various problems that arise in going from a model object to its image in the scene. An object may not appear the same in the scene as it was in the model, because it has undergone pose changes, or because it is occluded, or its colors appear different in the current illumination conditions. In addition, artifacts such as specularities, inter-reflections, shadows may also cause changes in the appearance of the object. So how can a model-driven selection mechanism meet these two apparently conflicting requirements? We now describe an approach to model-driven selection that meets some of these requirements. It makes a particular choice of model description and assumes that this is made available to it for selection. Since this model description affects the way our approach formulates the color-based model-driven selection problem, it is described first.

5.1 Model Description

The color region information in the model[3] (in

[3]The model description specifies a color view, that is, a

an image or view of the model, that is) is represented as a region adjacency graph (RAG) $M_G =< V_m, E_m, C_m, R_m, S_m, B_{rm}, S_{rm} >$, where V_m = color regions in the model, E_m = adjacencies between color regions, $C_m(u)$ = color of region $u \in V_m$, $R_m(u,v)$ = relative size of region 'v' w.r.t region u. $S_m(u)$ = size of region u, and B_{rm} = a bound on the relative size of regions given by R_m, and B_{sm} = a bound on the absolute size of regions given by S_m.

The above description exploits features of regions that tend to remain more or less invariant in most scenes where the model appears. If the color of a model region is specified by its color category, then as we discussed before, it tends to remain relatively stable (or changes in a predictable way) under variations in illumination conditions, and pose changes. Similarly, the adjacency information between two color regions tends to remain more or less invariant in the different appearances of the object, as long as the two regions are visible in the given image and there are no occlusions. Finally, the relative size of regions is preserved under changes of scale. But it can undergo considerable changes if the pose of the object changes, say when a region goes partially out of view. The bound on the relative size changes in each pair of adjacent region, B_{rm} indicates the extent of pose changes that a selection mechanism is expected to tolerate. Relative size changes can also occur due to occlusions. By placing some loose bounds on the absolute size changes as given by B_{sm}, the model description restricts the changes that can be tolerated in the presence of occlusions. For size changes in a region that go beyond the bounds, that region will be considered no longer recognizable, and then the selection will have to depend on the evidence for other model regions in the image.

This description is not very impoverished and has some structural information about color regions that can be used to restrict the number of false positives, and some constraints on the relative and absolute size changes that can be used to restrict the number of false negatives made by the selection mechanism.

Finally, the model description gives a way to analogously organize the color region information in the image as an image region adjacency graph as $I_G =< V_I, E_I, C_I, R_I, S_I >$, where each term has a meaning analogous to $< V_m, E_m, C_m, R_m, S_m >$ respectively.

5.2 Formulation of the Color-based Model-driven Selection Problem

In this section we will formulate the color-based model-driven selection problem as a type of subgraph matching problem. Given the image region adjacency graph, the model object if present in the scene represented in the image will form a subgraph in I_G. The location strategy can be regarded as the problem of searching for suitable subgraphs that satisfy the model description. Any such subgraph $I_g =< V_g, E_g, C_g, R_g, S_g >$ such that $\|V_g\| \leq \|V_m\|, \|E_g\| \leq \|E_m\|$, has associated with it a node correspondence vector $\Upsilon = \{(u_m, u_g) | u_m \in V_m, \forall u_g \in V_g \cup \{\perp\}, \{\perp\}$ is a null match$\}$. Although there are an exponential number of such subgraphs, not all of them correspond to model RAG. From the model description a set of unary and binary constraints could be derived (as is described later) that make only some subgraphs feasible. A feasible subgraph is, therefore, a subgraph that has all its nodes satisfying unary and binary constraints. For model-driven selection, since it is desirable to have at most one image subgraph matching the model RAG, we can select from among these subgraphs, a subgraph(s) that in some sense, best satisfies the model description. Here we formulate the color-based model-driven selection as the problem of choosing a feasible subgraph(s), I_g that minimizes the following measure:

$$\text{SCORE}(I_g) = (1 - \frac{\|V_g\|}{\|V_m\|}) + \frac{2 \sum_{\forall (u_g, v_g) \in E_g} R^2_{mg}(u_m, v_m, u_g, v_g)}{\|E_m\|} \quad (10)$$

where $R_{mg}(u_m, v_m, u_g, v_g)$ with $\Upsilon(u_m) = u_g$, $\Upsilon(v_m) = v_g$ expresses the change in the relative size when adjacent model regions (u_m, v_m) are paired to corresponding image regions (u_g, v_g) and is given by $R_{mg}(u_m, v_m, u_g, v_g) = \frac{|R_m(u_m, v_m) - R_g(u_g, v_g)|}{\max(R_m(u_m, v_m), R_g(u_g, v_g))}$. SCORE$(I_g)$ emphasizes rewards for making as many correspondences as possible as indicated by the first term, called Match(I_g), and penalties for a mismatch of the relative size, as indicated by the second term, called Deviation(I_g), which measures the mean square deviation of the relative sizes. Since the subgraphs are all feasible, the deviation accounts for occlusions and pose changes in a more refined way than the binary constraints alone. Another advantage of this measure is that it can be incrementally computed from individual region matches, so that a branch-and-bound search formulation can be used to reduce considerably the search involved in finding the best subgraph (i.e. the one with the lowest score). Finally, the above formulation is based on the hypothesis that at least one of the regions in the isolated subgraph corresponds to a model region. It is also designed primarily to locate single instances of the model object in the image. More instances can be found after removing the regions in the found instance from the image RAG.

5.3 A Color-based Model-driven Selection Mechanism

A color-based model-driven selection mechanism was built using the above formulation. The mechanism essentially uses a search strategy to find the best subgraph. The result of selection is the correspondence vector associated with the best subgraph. The search strategy used the following constraints to restrict the search among feasible subgraphs.

1.Unary constraints: The color and absolute region size information provided in the model description were used to develop unary constraints on these features. The

range of 2D views of the model in which one or more of the color regions described in the model are visible. If the model has some views showing an entirely different set of color regions, then they must be specified as separate color views.

color $C_g(u_g)$ of an image region u_g is said to match the color $C_m(u_m)$ on a model region u_m if these colors belong to same category or compatible categories (described in Section 2.4). With this scheme, brighter colors (of a given hue) in the model could potentially match to darker colors of the same overall hue in the image, thus accounting for simple lowering in illumination levels. The bounds on the absolute size provided by B_{sm} act as loose size constraint to rule out some clearly absurd scale changes (such as, say, a 100 fold increase in the smallest model region implying a blowup of the model outside the image bounds).

2. Binary constraints: The adjacency (as well as non-adjacency) and relative size information provided in the model were used as binary constraints to prune some impossible subgraphs. Specifically, the lack of adjacency in model regions is a powerful constraint, because two adjacent regions in the image cannot correspond to two regions that are not adjacent in the given color description (assuming a rigid model). Two adjacent regions in the model may, however, not appear adjacent in a given image due to occlusion. A simple analysis of occlusions could rule out several false matches in such cases (such as, say, discarding a match if the area spanned by the occlusion within a rectangle enclosing the candidate non-adjacent image regions far exceeds the combined size of the corresponding adjacent model regions). The bound on the relative sizes served as another binary constraint. The bound B_{rm} was used to constrain possible matches by requiring $R_{mg}(u_m, v_m, u_g, v_g) \leq B_{rm}(u_m, v_m)$.

3. Searching for the best subgraph

The search for the best subgraph (i.e. the subgraph that minimize the value of SCORE), can in principle, be done by an exhaustive enumeration of subgraphs. But with the algorithm described below, The search required is reduced to a large extent. The algorithm used is essentially a variation of the branch and bound interpretation tree (IT) search [4], with the major difference being that no verification is done when the search reaches a leaf node (as the task is selection and not recognition). Each level of the search tree represents a possible match for a model region (this includes a null match), so that the depth of the search tree is fixed by the number of nodes in the model RAG. The unary constraints are checked *a priori* to prune the breadth of the search tree. A subgraph in the image RAG that is potential match for the model RAG is represented by a path in the IT. The value of SCORE is updated at each node as $\text{SCORE}_{i+1} = \text{SCORE}_i - \frac{1}{\|V_m\|} + \frac{2R_m^2}{\|E_m\|}$. By keeping the lowest value of SCORE so far, search can be cut off below any node with a Deviation(I_g) value greater than the lowest SCORE value. In practice, the unary and binary constraints prune the search tree considerably so that the average number of full paths (up to the leaves) explored are few (≈ 50). Finally, after an instance of the model region has been found in the image, the selected area is removed and the search repeated on the resulting image RAG to look for more instances of the model object.

5.4 Results

The result of using color-based model-driven selection are illustrated in Figure 2. Figure 2a shows a model object, and its color description obtained by using the color-region segmentation algorithm of Section 3 is shown in Figure 2b. Here the background was removed by a simple threshold on intensities. This description is used to create a model RAG. Figure 2c shows a scene in which the model object occurs. The scene shown has several other objects with one or more of the model colors. Also, the model appears in a different pose here, being rotated to the left about the vertical axis. Figure 2d shows the result of applying the unary color constraints. The big blue glass matches the small blue flowers based on color alone. Next, the unary constraint on absurd size changes are used to prune the possibilities and the result is shown in Figure 2e. Finally, the subgraph with the lowest value of SCORE is shown in Figure 2f. As can be seen from this figure, a region containing most of the model object has been identified even though, the color image segmentation was not perfect (notice the a small streak above the white rim of the cup that merges with the book in the background).

5.5 Search Reduction using Color-based Model-driven Selection

The color-based model-driven selection mechanism provides a correspondence of model region to some image regions. The matching of model features to image features can be restricted to within corresponding regions, and this reduces the number of matches that need to be tried for recognition. To reduce the search further, conventional grouping can be performed within the selected color regions, as described in Section 4.2. To estimate the search reduction in this case, we continue with the analysis done in that section. Let N_l be the number of solution subgraphs given by the selection mechanism, and let I_k represent one such subgraph with the number of nodes = N_k. Let (G_{u_j}, G_{v_i}) = the number of groups in region u_j of the solution subgraph I_k, and region v_j of the model RAG that corresponds to u_j as implied by the correspondence vector Υ associated with I_k. Then assuming, as before, the average size of the group = g, the number of matches that need to be tried are $O(\sum_{k=1}^{N_l} \sum_{j=1}^{N_k} G_{u_j} G_{v_i} . g^3 . g^3)$. To compare this kind of selection with pure grouping we can take some typical values of these numbers. Letting M = 200, N = 3000, g = 7, G_M = 30, G_N = 430, G_{v_j} = 8, G_{u_i} = 5, N_l = 5, N_k = 5, we have the number of matches with grouping alone to be $O(G_M G_N g^3 g^3) \approx 1.56 * 10^9$, and using model-driven color-based selection with grouping, the number of matches become $\approx 1.25 * 10^8$. Assuming 1 microsecond as time per match this corresponds to reduction in match time from 26 minutes to $\approx$ 2 minutes. By trying several models and images of scenes where they occured, we recorded the average number of subgraphs generated by the model-driven selection mechanism. The search estimates were obtained using the above formula for model-driven selection with grouping, and the formulas for other methods mentioned in Section 4.2. The results are shown in Table II. The bound on

the number of groups in a region was the same as used in Section 4.2. As can be seen from the table, the number of matches using correspondence between model and image color regions is always lower. A curious feature to note from the table is that it takes less number of matches (and hence lesser time) for a more complex model (entry 1 in Table II) containing several color regions, than for a simple object with fewer regions (entry 2 in Table II). This is understandable since, with a large number of regions, the constraints are stronger and hence the false matches are fewer.

Discussion: The above studies estimated the search reduction without actually integrating the selection mechanism with a recognition system. Moreover , the estimated search was based on the assumption that there were no false negatives given by the selection mechanism. This can happen since a subgraph with the lowest value of SCORE may not always indicate a match to the model. To estimate the number of false positives, the number of false negatives, and the reduction in search that results due to this color-based selection mechanism, we have recently developed a 3D from 2D recognition system and are currently testing it. Preliminary results on using the selection mechanism as a front-end for recognition have so far been encouraging.

6. SUMMARY

In this paper we have shown how color can be used as a cue to perform both data and model-driven selection. Unlike other approaches to color, we have used the intended task to constrain the kind of color information to be extracted from images. This led to a fast color image segmentation algorithm based on perceptual categorisation of colors to given perceptually different color regions. This color description of the image formed the basis of data and model-driven selection. A saliency measure was then developed to rank the color regions to perform data-driven selection. Lastly, an approach to model-driven selection was presented that exploited description of model color regions to locate instances of model in the image. Finally, we regard color as one of the many cues that can be used for selection. Future research is directed towards using other cues such as texture to perform data and model-driven selection.

References

[1] D. Clemens, "Region-based feature interpretation for recognizing 3D models in 2D images," Ph.D Thesis, 1991.

[2] M.G. Engel, "Visual conspicuity and selective background interference in eccentric vision," Vision Research, vol.14, 1974, pp.459-471.

[3] J. Mahoney, "Image chunking: Defining spatial building blocks for scene analysis," Technical Rep. TR-980, Artificial Intell. Lab., MIT, 1986.

[4] W.E.L.Grimson., *Object Recognition by Computer: The Role of Geometric Constraints*, MIT Press: Cambridge, 1990.

[5] B.K.P.Horn, *Robot Vision*, MIT Press: Cambridge, 1986, Chapter 4.

[6] D. Huttenlocher and S. Ullman, "Object recognition using alignment," First Int. Conf. Computer Vision, London, 1987, pp.102-111.

[7] D.W. Jacobs, "Grouping for recognition," MIT AI Lab Memo. 1177, 1989.

[8] D.G. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic: Boston, 1985.

[9] A. Hurlbert, *The Computation of Color*, Tech. Rep. TR-154, Artificial Intell. Lab., MIT, 1989.

[10] E. Kandel and J. Schwartz, *Principles of Neural Science*, Elsevier: New York, 1985, Chapter 30, pp.386-388.

[11] G.J. Klinker, S.A. Shafer, and T. Kanade, "A physical approach to color image understanding," Intl. Jl. Computer Vision, vol.4, no.1., pp.7-38, Jan. 1990.

[12] E. Land, " Recent advances in retinex theory," in *Central and Peripheral Mechanisms of Colour Vision*, T. Ottoson and S. Zeki Ed., pp. 5-17, London:McMillan, 1985.

[13] L.T. Maloney and B. Wandel, "Color constancy: A method for recovering surface spectral reflectance," Jl. Optical Society of America, vol.3, 1986, pp.29-33.

[14] G.W. Meyer and D.P. Greenberg, "Perceptual color spaces for computer graphics," Computer Graphics, vol.14, no.3, pp. 254-261, July 1980.

[15] D.Sagi and B. Julesz, ""where" and "what" in vision", Science, vol.228, 1985, pp.1217-1219.

[16] A. Shashua and S. Ullman, "Structural saliency : The detection of globally salient structures using a locally connected network," in Proc. 2nd Int. Conf. Computer Vision, pp. 321-327, Florida, Dec. 1988.

[17] E. Sternheim and R. Boynton, "Uniqueness of perceived hues investigated with a continuous judgemental technique," Jl. of Experimental Psychology, vol.72, pp.770-776, 1966.

[18] M.J. Swain and D. Ballard, "Indexing via color histograms," Third Int. Conf. Computer Vision, 1990.

[19] L.E. Wickson and D. Ballard, "Real-time detection of multi-colored objects," in SPIE Sensor II: Human and Machine Strategies, vol.1198, Nov. 1989.

[20] G. Wyszecki and W.S. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae*, John Wiley & Sons, New York, NY, 1982.

[21] J.D. Foley and A. Van Dam, " *Fundamentals of Interactive Computer Graphics*, Reading: Addison Wesley, 1984.

[22] D.T. Clemens and D.W. Jacobs, "Space and time bounds on indexing 3D models from 2D images," to appear in IEEE Trans. Pattern Anal. and Machine Intelligence, 1991.

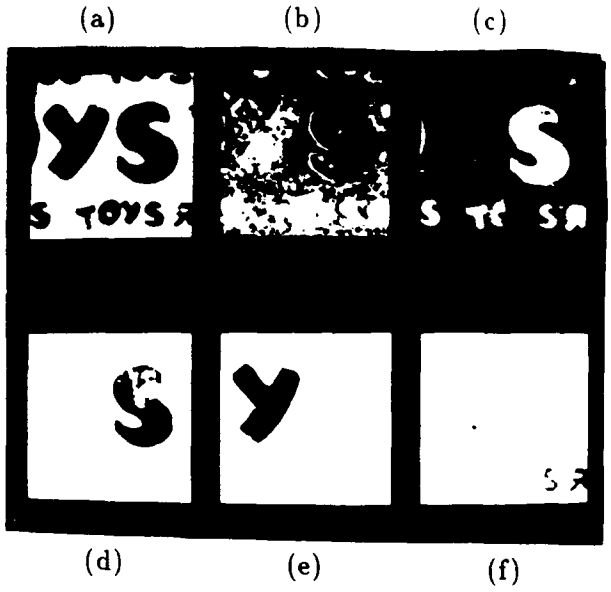


Figure 3: *Illustration of color region segmentation and color-saliency. (a) Input image consisting of regions of 3 different colors: red, green and blue against an almost white background. (b) Result of Step 2 of algorithm with regions colored differently from the original image. (c) Final segmentation of the image of Fig.3a. (d) — (f) The three most distinctive regions found using the color saliency measure.*

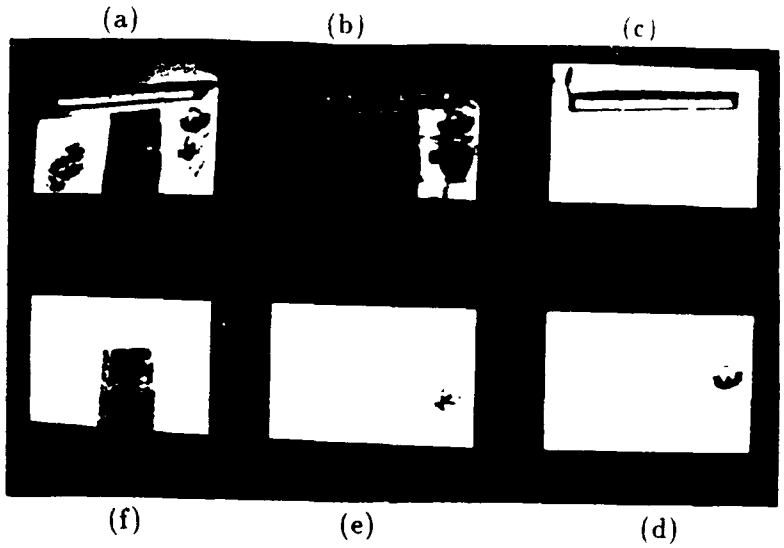


Figure 4: *Illustration of color region segmentation and color-saliency — Last example. (a) Input image depicting a scene of different kinds of objects (cloths and polished book). (b) The color regions extracted from (a) using the color region segmentation algorithm. (c)-(f) The four most distinctive regions detected using the color-saliency measure.*

S.No	M	N	M_R	N_R	No selection		Only grouping		Salient color + grouping	
					Num. matches	Time	Num. matches	Time	Num. matches	Time
1.	229	1170	1	18	$1.92\text{x}10^{16}$	610yrs	$6.52\text{x}10^{8}$	11min	$3.37\text{x}10^{8}$	5min
2.	507	2655	2	20	$2.4\text{x}10^{18}$	77,341yrs	$3.22\text{x}10^{9}$	54min	$1.32\text{x}10^{9}$	22min
3.	124	2655	2	20	$3.57\text{x}10^{16}$	1131yrs	$8.05\text{x}10^{8}$	13min	$3.3\text{x}10^{8}$	5min
4.	507	2247	2	14	$1.48\text{x}10^{18}$	46,884yrs	$2.72\text{x}10^{9}$	46min	$7.8\text{x}10^{8}$	13min

Table I: Search reduction using color-based data-driven selection. The last column shows the match time when color-based data-driven selection is combined with grouping. The color-based selection is done by choosing the four most salient regions. Here g = 7, Time per match = 1 microsecond, and the grouping method is as described in text.

S.No	M	N	M_R	N_R	N_l	N_k	No selection		Only grouping		Model-driven selection	
							Num. matches	Time	Num. matches	Time	Num. matches	Time
1.	786	3268	5	30	1	(3)	$1.69\text{x}10^{9}$	530000yrs	$6.15\text{x}10^{9}$	103min	$4.55\text{x}10^{7}$	45sec
2.	83	3078	1	20	3	(1,1,1)	$1.67\text{x}10^{16}$	528yrs	$6.2\text{x}10^{8}$	11min	$1.7\text{x}10^{8}$	3min
3.	507	2655	2	20	2	(2,1)	$2.4\text{x}10^{18}$	77,341yrs	$3.22\text{x}10^{9}$	54min	$3.72\text{x}10^{8}$	6min
4.	507	2247	2	14	1	(2)	$1.48\text{x}10^{18}$	46,884yrs	$2.72\text{x}10^{9}$	46min	$3.16\text{x}10^{8}$	5min

Table II: Search reduction using color-based model-driven selection. The last column shows the match time when model-color-based selection is combined with grouping. Here g = 7, Time per match = 1 microsecond, and the grouping method is as described in text.

Space Efficient 3D Model Indexing

David W. Jacobs
MIT AI Lab, 545 Technology Square, Room 751
Cambridge, MA 02139

Abstract

We show that the set of 2D images produced by the point features of a rigid 3D model can be represented with two lines in two high-dimensional spaces. These lines are the lowest-dimensional representation possible. We use this result to build a system for representing in a hash table at compile time, all the images that groups of model features can produce. Then at run time a group of image features can access the table and find all model groups that could match it. This table is efficient in terms of space, and is built and accessed through analytic methods that account for the effect of sensing error. In real images, it reduces the set of potential matches by a factor of several thousand. We also use this representation of a model's images to analyze two other approaches to recognition: invariants, and non-accidental properties. These are properties of images that some models always produce, and all other models either never produce (invariants) or almost never produce (non-accidental properties). In several domains we determine when invariants exist. In general we show that there are an infinite set of non-accidental properties that are qualitatively similar.[1]

1 Introduction

Object recognition systems typically search for matches between image features and model features that are consistent with some transformation of the model into the image. This search can require a great deal of computation, particularly in challenging domains such as the recognition of 3D objects from cluttered 2D images, and the identification of an object from a large data base of possible objects. One approach to handling this complexity is to decompose the recognition task so that as much of the work as possible is done on the models alone, at compile time, and on the image alone, in a bottom up process. The results of these two separate computations are then combined in a simple comparison.

Such an approach to recognition can take the form of indexing, combined with grouping. By indexing, we mean that the system places pointers to groups of model features in a hash table, at compile time. At run time, a group of image features accesses the hash table to find all the model groups that might match it. If we want efficient run-time processing, image groups will each access the table in only a single place. That means that the table must, in some form, represent the entire set of images that each model group could produce. This paper determines the most space-efficient possible method for representing a 3D model's point features in such a hash table. It also presents a method for analytically determining which entries to make in such a table.

Indexing may be combined with grouping. Grouping selects a relatively small set of groups of model features to enter in the hash table. It then selects a small number of groups of image features in a model-independent, bottom up process. This avoids the combinatoric explosion that occurs when all possible image groups and model groups are considered. With effective grouping, large groups become desirable input to the indexing system, because they provide the greatest discriminatory power. That is, fewer model groups will match a large image group than a small image group. See [4] for further discussion of this point. In this paper we focus on the problem of building an indexing system appropriate for use with grouping.

The central problem of indexing is to determine the most economical possible representation for the set of image groups that each model group might produce. It is trivial to do indexing by making a different entry in the hash table for every different image group a model group can produce, but this would require excessive space. So instead we seek a representation of images such that minimal space is required to describe all these images.

In 2D domains this has been done with invariant descriptions, that is, with descriptions of images that have the property that every image of a given model group produces the same description. Invariants have been found that can capture all the information in the image group that can be used to match it to model groups, while allowing us to represent each model group with

[1]Support for this research was provided in part by the University Research Initiative under Office of Naval Research contract N00014-86-K-0685, and in part by the Advanced Research Projects Agency under Army contract DACA76-85-C-0010 and under Office of Naval Research contract N00014-85-K-0124.

a single entry in the indexing hash table. However, it has been shown that such invariants do not exist for the description of 3D models[3], [4], [16]. Furthermore, [4] shows that to perform indexing of 3D models, each model must be represented by a 2D surface in a single index space. Unfortunately, large amounts of space are required to accurately represent such a 2D surface discretely. In this paper, however, we show that the images of a model group can be represented as a 2D surface that can be canonically decomposed into two 1D surfaces, represented in two orthogonal index spaces. This result makes the space requirements of indexing reasonable, because we can discretize two 1D surfaces using much less space than would be required to discretize one 2D surface.

Our results also provide a simple conceptualization of the recognition problem, which we use to analyze two other approaches to recognition. First, when we constrain our universe of objects to contain only some collections of 3D points, the question resurfaces as to whether invariant functions exist. We use our previously stated result to answer this question in several domains. We also consider a second, related approach to recognition. This approach uses non-accidental properties, which are defined as properties of an image that some models produce from all viewpoints, and other objects only produce from almost no viewpoints. A small set of non-accidental properties have been used. We show that they are a few instances of an infinite class of such properties, all of which are qualitatively similar.

Another problem left unanswered in previous 3D indexing systems was the analytic computation of the entries to make in the hash table. Previously, this has been done only by sampling the set of possible viewpoints of each model. In this paper we present an analytic method of building the indexing table, which we combine with an analytic method of accounting for the effects of sensing error. These results have allowed us to build a practical indexing system, which we demonstrate using real images. The resulting indexing system produces speedups of up to a factor of several thousand over brute force search.

2 Describing the Images a Model Produces

This section describes a compact, analytically determined representation for the set of all images that a general 3D model may produce. Models are assumed to consist of any arbitrary collection of ordered 3D point features. We ignore sensing error when describing the model because we account for error with regard to a particular image of the model, at lookup time. We are then able to describe all images of a model with two straight lines located in two orthogonal spaces. These lines can easily be derived analytically from the model. This representation is optimal in the dimensionality of the space required, and in introducing no false positive or false negative information.

In describing the images that a model produces we use the following novel model of projection. First, we assume that the object is imaged from an arbitrary viewpoint using orthographic projection with scale. Orthographic projection with scale is a common approximation to perspective projection. Next, we allow an arbitrary affine transform to be applied to the resulting image. Applying an arbitrary affine transform to an image is equivalent to viewing that image from an arbitrary position, assuming that this projection also is a scaled orthographic projection. Therefore, our projection model encompasses all images that a model might produce, as well as all images that a photograph of the model might produce.

There are several reasons for using this projection model. As we will see, it is mathematically convenient. But in addition, it allows us to build an indexing system that can recognize photographs of objects. This also suggests the hypothesis that human ability to interpret photographs is an epiphenomenon of the fact that, for computational reasons, our visual system does not make use of features of an image that vary under affine transforms (see [6] for discussion of a related hypothesis). Finally, it may be easily shown that this model of projection is equivalent to multiplying 3D model points by an arbitrary two by three matrix, and then adding an arbitrary translation vector. This seems to be the simplest linear projection model from 3D to 2D (see [19] for further discussion of this projection model).

We now show that under this model of projection, the set of images produced by any model is described by the cross product of two lines in two orthogonal spaces. To do this, we represent images as follows. As an image consists of 2D point features, we use the first three points to define an affine basis. That is, if we denote the image points: $(p_1, p_2, ..., p_n)$, let:

$$\mathbf{o} = \mathbf{p_1} \qquad \mathbf{u} = \mathbf{p_2} - \mathbf{p_1} \qquad \mathbf{v} = \mathbf{p_3} - \mathbf{p_1}$$

Then we may fully describe the locations of the remaining points using affine coordinates derived with respect to this basis. For example, we describe p_4 with the parameters (α_4, β_4), where:

$$\mathbf{p_4} = \mathbf{o} + \alpha_4 \mathbf{u} + \beta_4 \mathbf{v}$$

Then an image is fully described by the parameters: $(\mathbf{o}, \mathbf{u}, \mathbf{v}, (\alpha_4, \beta_4), ...(\alpha_n, \beta_n))$. It is important to what follows that the affine coordinates of a point are left unchanged by any affine transform ([13]).

Due to the model of projection we use, we may ignore the first three of these parameters. To see this, we note that, except in degenerate cases, there exists an affine transform that will map any three image points to any other three image points. Therefore, under the type of projection that we consider, if a model can produce the image, $(\mathbf{o}, \mathbf{u}, \mathbf{v}, (\alpha_4, \beta_4), ...(\alpha_n, \beta_n))$, it can also produce the image $(\mathbf{o'}, \mathbf{u'}, \mathbf{v'}, (\alpha_4, \beta_4), ...(\alpha_n, \beta_n))$ for any choice of $(\mathbf{o'}, \mathbf{u'}, \mathbf{v'})$, by combining the affine transform that maps $(\mathbf{o}, \mathbf{u}, \mathbf{v})$ to $(\mathbf{o'}, \mathbf{u'}, \mathbf{v'})$ with the affine transform that was part of the projection that produced the original image. Therefore, the parameters $(\mathbf{o}, \mathbf{u}, \mathbf{v})$ provide no information about whether a model could produce an image.

The remaining image parameters form what we will call an *affine space*. An image with n ordered points is mapped into a point in a $2(n-3)$-dimensional affine space

by finding the affine coordinates of the image points, using the first three as a basis. We divide the affine space into two orthogonal subspaces, an *α-space*, and a *β-space*. The *α-space* is the set of α coordinates of the image's affine coordinates, and the β-space is similarly defined. The affine space is then equal to the cross product of the *α-space* and the *β-space*, and each image corresponds to a point in each of these two spaces. The previous paragraph states that the images that a model can produce are fully described by the locus of points these images map to in affine space. We now show, that for any model, these images map to the cross product of lines in *α-space* and *β-space*.

First we must note that our model of projection can be decomposed into one part, which captures the effects of viewing direction, and a second part, which is an affine transform of the projected image. We can think of orthographic projection as projecting the model along some viewing direction into a plane. Following this, orthographic projection also allows rotation in the plane, translation, and scale, but we can ignore these parts of the transformation by folding them into the affine transformation that we allow on the image, following the orthographic projection. To find the parts of affine space corresponding to images of a particular model, we only need to consider the sets of affine coordinates that a model may produce in an image as the viewing direction varies. The subsequent affine transformation leaves the affine coordinates unchanged.

We now assume the model consists of at least five points. Call the plane determined by the first three model points, the *model plane*. If we project the fourth model point, m_4, perpendicularly into the model plane, we call this point m'_4. Since m'_4 is in the plane of the first three model points, we can discuss its affine coordinates with respect to these three model points. We call these affine coordinates (a_4, b_4). Similarly, for the j'th model point, m_j, we define m'_j and (a_j, b_j) (see figure 1). Without loss of generality, assume the model plane is $z = 0$. So z_4 is the height of m_4 above this plane, and z_j is the height of m_j above the model plane. We define $r_j = \frac{z_4}{z_j}$.

We now show that for any affine coordinates (α_4, β_4), there is a viewpoint in which the projection of m_4 has those affine coordinates. We then express the affine coordinates of the remaining projected model points as a function of (α_4, β_4). Some point, i'_4, in the model plane has affine coordinates (α_4, β_4). If we form a line including i'_4 and m_4, this line describes a viewing direction from which m_4 and i'_4 project to the same image point, i_4. Since i'_4 is coplanar with the first three model points, it has the same affine coordinates when viewed from any direction, since affine coordinates of planar points are not changed by any affine transformation, and viewing a planar object from an arbitrary viewpoint is equivalent to applying an affine transform. So i_4 has affine coordinates (α_4, β_4).

A line parallel to the viewing direction will also pass through m_j, and intersect the model plane at a point we call i'_j. The affine coordinates of i_j, the image of m_j, are the same as the affine coordinates of i'_j in the model

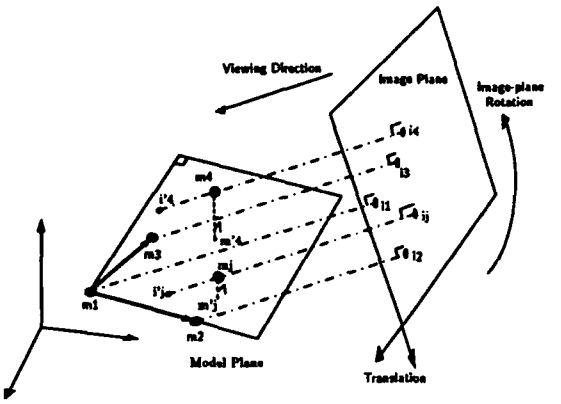


Figure 1: The image points i_1, i_2, i_3, i_4, and i_j are the projections of the model points m_1, m_2, m_3, m_4, and m_j, before the affine transform portion of the projection is applied. The values of the image points depend on the pose of the model relative to the image plane. In the viewing direction shown, i'_4 and m_4 project to the same image point. m'_4 is in the model plane, directly below m_4. Note that i_4 has the same affine coordinates as i'_4.

plane. Since the line connecting m_j to i'_j is parallel to the line connecting m_4 to i'_4, the triangle $m_4 m'_4 i'_4$ will be similar to the triangle $m_j m'_j i'_j$, and scaled by a factor of r_j. In particular, this means that: $(i'_4 - m'_4) = r_j(i'_j - m'_j)$, and therefore:

$$(\alpha_j, \beta_j) = (a_j, b_j) + \frac{((\alpha_4, \beta_4) - (a_4, b_4))}{r_j}. \qquad (1)$$

This equation describes all image parameters that these five points may produce. For any image, this equation will hold. And for any values described by the equation, there is a corresponding image that the model may produce, since, from above we know that for any values (α_4, β_4), there is a view of m_4 that produces these values. Taking the α component of these equations we have equations that describe a line in α-space. We may derive a similar set of equations in β-space. These equations are independent. That is, for any set of α coordinates that a model may produce in an image, it may still produce any feasible set of β coordinates. There are also degenerate case. If some of the model points are coplanar, than some of the r_j are infinite, and the line is vertical in those dimensions. If all the model points are coplanar, the affine coordinates of the projected model points are invariant, and each model is represented by a point in affine space. If the three model points are colinear, then the line is undefined.

Notice that for any line in α-space, there is some model whose images are described by that line. It is not true that there is a model corresponding to any pair of lines in α-space and β-space because the parameters r_j are the same in the equations for the two lines. This means that the two lines are constrained to have the same directional vector, but they are not further constrained.

In the absence of image error, indexing could proceed

in the following way. Form two hash tables which discretize α-space and β-space. For each model, compute the corresponding line in each of these spaces, and make an entry in each bucket that intersects one of these lines. At run time, given an image, determine the points in α-space and β-space that correspond to that image. Look in the appropriate bucket in each of the two spaces, and intersect the results. If the buckets are made sufficiently small, this will produce exactly the set of models that could produce the image.

To account for error, for a particular image we must determine the volumes of α-space and β-space that might match that image when each possible set of errors is assumed. In [12] and [9] we solved this problem for images consisting of four points. That solution tells us the range of affine coordinates that a fourth image point might have with respect to the other three, when we allow for bounded sensing error. By applying that solution to each image point in turn, we find bounds on the set of possible affine coordinates that an image might produce. This method is conservative, in that it overstates the effects of error in two ways. The effect of error on the α and β coordinates is not independent, but we separately determine the ranges of possible α and β values that can occur. Secondly, we treat the error as if it had an independent effect on each pair of affine coordinates. However, error in the three basis points has a related effect on the affine coordinates of every other point. In effect, we enclose the true volume of affine space consistent with error inside the smallest possible rectanguloid that has sides parallel to the axes of the affine space. While this may produce some unnecessary matches, it can not cause us to miss any correct matches.

Representing each model group's images with two 1D surfaces is an important improvement over using a single 2D surface, because it requires much less space to discretely represent a pair of 1D surfaces than to discretely represent a 2D surface. There is a run-time price that must be paid for this space, when we intersect the results of two separate lookups. However, this cost is negligible in the actually system we have built. We note that the dimensionality of the surfaces used to represent a model group's images is now the best that can be done, since [4] has shown that in a single index space, a 2D surface is required to represent these images, and it is not possible to represent a 2D surface as the cross-product of any countable number of countable sets.

3 Invariants and Non-Accidental Properties

Before describing this indexing system further, we will consider some implications of this view of the recognition problem. In the error free case, we have almost entirely reduced the recognition problem to a very simple form, in which recognition is the problem of determining which points fall on which lines. To demonstrate the usefulness of this, we will consider two influential approaches to recognition from this point of view.

A number of recognition systems have been based on *invariants*. In the context of recognition, an invariant is a function of the image that has the following property: if f is an invariant function, then for any model, m, if i_1 and i_2 are images of m, then $f(i_1) = f(i_2)$. That is, an invariant is a property that is true of all images of a model, and hence does not vary under the transformation that turns a model into an image.

2D recognition systems have long implicitly relied on the descriptions of 2D objects that do not vary as the object is rotated or translated in the plane. More recently, invariants have been used for the recognition of planar models from arbitrary, 3D views ([22], [13], [14], [7], [8], [20]). Recently, [3], [4], and [16] have proven that there are no non-trivial invariants when models may consist of arbitrary collections of point features. These proofs took the following form. Given any two models, M_1 and M_2, a set of intermediate models, P_1, P_2 ... P_n were constructed. By this construction, M_1 and P_1 produce a common image, hence any invariant function would have to have the same value on any images these models produce. Similarly, P_1 and P_2 produce a common image, and so on, until M_1 and M_2 are linked by this series of intermediate models. Hence, the invariant function must produce the same value for images of any two models, and is trivial.

Our formulation of the recognition problem makes it easy to prove a number of results about invariants. These results will only apply to our model of projection. However, we note that considering our projection model is equivalent to assuming that any invariant function will be both invariant for the orthographic projection of a set of 3D models, and will also be invariant for the orthographic projection of 2D images of these models.

We can now show that there are no invariant functions of 3D models with a proof that requires only two intermediate models. Suppose model M_1 corresponds to the two lines, A_1 and B_1 in α-space and β-space respectively. Similarly, suppose model M_2 corresponds to A_2 and B_2. Then there are an infinite number of lines that intersect both A_1 and A_2. Choose one of these, A_1'. Choose B_1' as any line that is parallel to A_1', and intersects B_1. Then there is a model, P_1 that corresponds to the lines (A_1', B_1'). P_1 has an image in common with M_1, since A_1 intersects A_1', and B_1 intersects B_1'. We may then construct P_2 and its lines, (A_2', B_2'), so that B_2' intersects B_1' and B_2, and so that A_2' passes through the point where A_1' and A_2 intersect. So P_2 will have an image in common with P_1 and M_2. Therefore, any invariant function must have the same value for any image of any of the four models.

We may also use our previous results to examine other questions about the occurrence of invariants. As [16] points out, there may be invariant functions for a particular set of 3D models, if the models can be divided into non-trivial equivalence classes, where two models are equivalent if they have an image in common, or are both equivalent to another model. From our previous work, it becomes easy to form these equivalence classes for a particular set of models, because we can tell that two models produce a common image if their corresponding lines in α-space and in β-space intersect.

Another question that arises for a restricted set of

models is whether there is an invariant function that will produce no false positive matches. We can think of an invariant function as assigning values to models, because it assigns the same value to every image of a model. An invariant function leads to no false positives if the function maps a model and an image to the same value only when there really is a transformation that will cause the model to produce that image.

To answer this question we consult our representation of a model's images as planes in affine space, that is, we take the cross product of the model's pair of lines in α-space and β-space. We can see that there is an invariant function with no false positives for a specific set of models if and only if no two planes that correspond to two models intersect without completely coinciding. If this condition is met, we may construct an invariant function that assigns a common value to all the images that fall on a model's plane of images, and assigns a different value to any two images that lie on the planes of different models. Then, any model that can produce one image with a particular value of the invariant function will be able to produce exactly the set of images that have that value of the invariant function. If, on the other hand, two planes intersect and do not coincide, then all images that either plane contains must have the same value for any invariant function, but neither object can produce all these images, so false positives will occur.

From this result we can see that a specific set of models can have an invariant function with no false positives only if it is a measure 0 subset of the set of all possible models. There is a correspondence between the set of all possible models and the set of all planes in affine space, subject to the restriction that for each plane, the directional vectors of the two lines it produces when projected onto α-space or β-space must be the same. Any set of such planes that do not intersect is a measure 0 subset of the set of all such planes. For example, if we have a restricted set of models corresponding to a set of non-intersecting planes in affine space, this means that any point in affine space can belong to only one of these planes, although it belongs to uncountably many planes that correspond to some model.

An especially interesting special case is that of models containing five points. This is the smallest group that can produce invariants, because in general any four model points can appear as any four image points. Most systems based on invariants have used the smallest possible model groups, in order to limit the number of possible model groups they must consider (this is true of [13], [8], [20], for example). A set of model groups of five points each will each produce a pair of lines in 2D α-space and β-space. Furthermore, recall that these two lines will have the same slope, which means that two different models will produce lines that are either parallel in both spaces, or that intersect in both spaces. Therefore, an invariant function for groups of five points is possible only when all lines produced by all models are parallel. For example, if one model produces lines not parallel to the others, it will have an image in common with each of them, implying that the invariant function must be constant over all images produced by all models. The

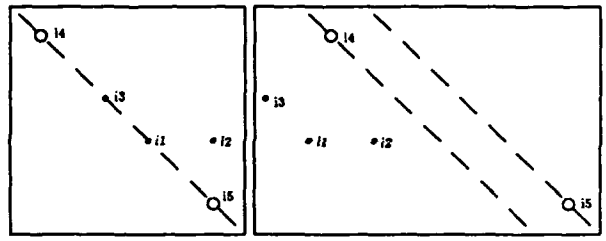


Figure 2: If an image corresponds to the point $(0,0)$ in α-space, that means that the last two points in the image are colinear with the first and the third points. This is shown on the left, where the first three points are shown as dots, and the second two points, shown as open circles, must lie somewhere on the dashed line. If the image corresponds to $(2,3)$ in α-space, the points must fall on the two lines shown on the right. In both cases we have an equivalent non-accidental property.

lines produced by all models will be parallel only when r_5, (see equation 1), is the same for all models, that is, when the ratio of the height above the model plane of the fourth point to the height of the fifth point is always the same.

We now turn to the analysis of some non-accidental properties. Lowe[15] first used these in his recognition system, SCERPO, and Biederman[1] has based his GEON approach to recognition on these properties. A non-accidental property is a property of an image such that some models only produce images with that property, while for other models either no or almost no images they produce have that property. The work of Lowe and Biederman has been based on identifying a small number of such properties, such as parallelism, colinearity, or symmetry. Our above work now allows us to see that there is an infinite class of such properties, and that there is nothing about the inherent geometry of models which makes one such property qualitatively different from another.

First, we illustrate this in the simplest case of colinearity. We will show that there are an infinite set of properties that are qualitatively similar to colinearity. That is, if we ignore the distribution of models in our universe of possible models, there is nothing we can say about colinearity that we can not also say about an infinite number of other properties. Suppose we have an image group with five points, and the first point and last three points are colinear. This is equivalent to saying that α_4 and α_5 of the image's affine coordinates both equal 0. That is, any image which corresponds to the point $(0,0)$ in α-space, and to any point in β-space, has those four points colinear. If we look at this from the perspective of non-accidental properties, we say that either the four points are colinear in the model that produced them, in which case they always appear colinear, or they are not really colinear, in which case there is only a measure 0 chance that they would appear as colinear. We would then infer that we should match these colinear image points with only similarly colinear model points.

From our new perspective, we would say that there are two possibilities. Either the model that produced

this image is planar, and always produces images with the values $(0,0)$ in α-space, or the model corresponds to a line in α-space that passes through the origin. These two views of the situation are equivalent. However, we may now see that there is nothing special about the coordinates $(0,0)$ in α-space. We could say the same thing about any other coordinates. For example, we get a similar non-accidental property when we consider an image with affine coordinates $(2,3)$ in α-space (see figure 2). Again, either the object that produces such an image is planar, and always produces such an image, or the object corresponds to a line in α-space that passes through the point $(2,3)$. So the α-coordinates $(2,3)$, and any other pair of α-coordinates, define a new non-accidental property, and there is no qualitative difference between these properties and colinearity that can be inferred from the fundamental geometry of objects.

More generally, other non-accidental properties, such as parallelism or skew symmetry, can also be thought of as describing regions in affine space. The non-accidentalness of these properties is equivalent to saying that a planar object that produces these properties from one viewpoint will always produce them, while a non-planar object that can produce the property will be equivalent to a pair of lines that intersects the property in only a measure 0 portion of its extent. If two properties correspond to similarly shaped regions in affine space, then there will be no qualitative difference between them.

This argument does not imply that non-accidental properties are not useful, or that the non-accidental properties that have been proposed are not more useful than others. It simply suggests that the special perceptual saliency of some non-accidental properties, such as colinearity, lies with the particular nature of objects that occur in the world, and can not be explained by some qualitatively different characteristic of the geometry of these properties.

4 Determining the Projected Model

Our indexing system produces matches between groups of image features and groups of model features. While there are standard techniques for verifying such hypotheses, our representation of models lends itself to a particularly simple method of determining the projection of the model into the image.

A match produced by the indexing system provides us with lines in α-space and β-space that represent the model, and a pair of points in these spaces for the image. By finding the locations on the lines that are closest to these two points, we find the error-free set of affine coordinates that best fit our image. We then can use any one of these coordinates to determine the affine coordinates of any point on the projected model, since, as we pointed out above, all further affine coordinates are functions of any pair of affine coordinates.

Let us illustrate this with an example. Suppose indexing matches model points m_1, m_2, m_3, m_4, m_5, m_6 to image points $i_1, i_2, i_3, i_4, i_5, i_6$, and suppose based on this match we wish to project model points $m_1, m_2, ...m_n$ into the image. At compile time, we use the points m_1, m_2, m_3 as a basis, and compute the two lines in the affine spaces that describe all images of the model when the image of these three points are used as a basis. (This is done at compile time for all possible basis triples). Call these two lines L_1 and L_2. These lines are in $(n-3)$-dimensional affine spaces, $(\alpha_4...\alpha_n)$ and $(\beta_4...\beta_n)$, because they represent the locations of $n-3$ points using the first three points as a basis. Our six matched image points map to two points in the 3-dimensional affine spaces $(\alpha_4, \alpha_5, \alpha_6)$ and $(\beta_4, \beta_5, \beta_6)$. Call these points p_1 and p_2. By projecting L_1 and L_2 into these lower dimensional spaces, we get lines that describe the possible images that the first six model points can create. By finding the point on the projection of L_1 closest to p_1 we find the α coordinates of the image of the model that best match the image points. Similarly, we find the appropriate β values. These values determine locations on L_1 and L_2 that tell us the affine coordinates of all the model points in the image that will best fit the matched image points. Without explicitly computing the viewing direction we have computed the appearance of the model, when seen from the correct viewing direction. (A different method must be used if the matched model points are coplanar, because in that case their affine coordinates provide no information about viewing direction).

In addition to determining the effects of the viewing direction on the image, we must also allow for the effects of the affine transformation portion of the projection. However, once we have determined the affine coordinates of all the projected model points, it is straightforward to apply a least squares method to find the affine transformation that optimally aligns the image points with the projected model points.

5 Experiments with an Indexing System

5.1 The Recognition System

Our indexing system is designed to match relatively large groups of image points to equally large groups of model points. Such an approach requires a bottom-up grouping process to control the combinatorics of forming all possible large groups of image and model points. While we are currently interfacing this system with a particular grouping system, our purpose here is to examine the performance of the indexing system alone. So that our results will be independent of the deficiencies of any specific grouping system, we have tested the indexing system with somewhat ideal groups, in which automatically located features are formed into groups by hand.

We begin model building by running an edge detector on many images of the object. We find corner features by making straight line approximations to the edges, and locating a corner where nearby lines have a stable intersection point, when extended. We then form by hand groups of three to five points that are formed by a set of convex lines (see [11] and [10] for discussion of the value of convex groups). The convexity of the group orders the points, although it does not tell us which point comes first. So a different group is formed for each possible starting point. To allow for the effects of occlu-

sion, we also form groups in which any one of the points is omitted, as long as the group still contains at least three corners. Corners that appear in these groups are matched by hand between the images used to build a model.

Since three to five points do not provide sufficient information to discriminate between models, we then form all pairs of these groups. Each pair of groups gives us an ordered set of points. For each set of points, we calculate the lines in α-space and β-space to which they correspond. Each image that contains all the corners appearing in the group is used to calculate two points, one in α-space and one in β-space, that describe the ordered group. We then fit lines to these points, to determine the set of all images that the group could produce.

We then compute which buckets of discretized α-space and β-space these lines intersect, and make an entry in each bucket, pointing to the appropriate group of model points.

To build a model of the object's line segments we match by hand corners that are at the end points of line segments that appear in a number of images of the model. Then, for every triple of corners used as a basis for one of the groups entered into the lookup table, we form lines in the two affine spaces representing the possible affine coordinates of the corners of all the model's line segments. We use a new hash table to store these pairs of lines for easy access.

At run time, we take a new picture of the model, along with occluding objects. We form groups from the corners of this picture, just as we did for the images used to build the model. These groups may be missing some of the corners that appear in the modeled groups, due to occlusion or due to failure in the corner finder. Next, as before, we form all pairs of these groups. We have some freedom in how we order the points in these pairs of groups before using them for lookup. In building the model, each point in a group was used as a starting point for that group. That point, and the next two, were used as a basis for computing the affine coordinates of the remaining points. Since we may use any point in the image group as a starting point, we select the point which gives us the most stable set of basis points. Then, we use this ordered set of image points to compute two rectanguloids in the affine spaces, as described above, and find all matching sets of model points. Since we can only represent a finite portion of the affine spaces, it is possible that the rectanguloids produced by a group will fall outside the bounds of this portion of affine space. We ignore such groups, and in fact, groups that produce large affine coordinates are likely to be unstable, providing poor candidates for matching.

We perform this indexing for each pair of image groups. Ordering these pairs so that we start with the pair that produces the fewest matches, we then perform verification on each match until the object is found. We use the method described above to determine the projection of the model's line segments for each match. We then search the image for line segments that are near the model's projected line segments, and that have roughly the same orientation, in order to determine the fraction of the model which the image can explain. We stop when a sufficiently good hypothesis is found.

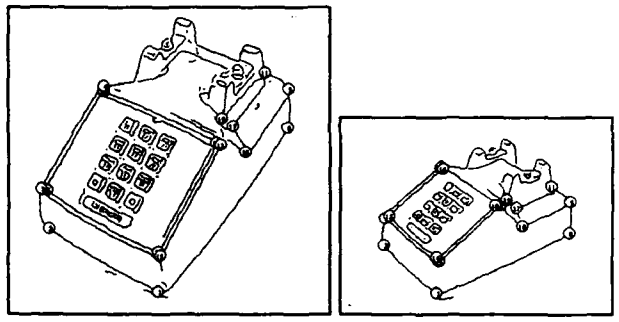


Figure 3: Edges from two of the pictures used to build a model of the phone. Circles indicate the location of automatically found corners that were then selected by hand to be in the model. They are numbered for reference.

Note that in general, when determining the appearance of a model from a given viewpoint, we should eliminate lines that are not visible from that viewpoint. To avoid the need for this and simplify our verification system, we have taken all images from a single aspect of the object. That is, we restricted our viewpoint to about a quarter of the viewing hemisphere, in which all the same set of points and lines were visible.

5.2 Experiments

In experimenting with the above recognition system, we have used the following values for various parameters. We allowed image error of five pixels in indexing. The index table represents all affine coordinates between -25 and 25. Each dimension of the table is divided into 100 intervals. We make the buckets of uniform size for affine coordinates between 0 and 1, and then increase the bucket size linearly for coordinates above 1 or below 0. We do this because error has a greater effect on images with higher affine coordinates, at approximately this rate. A projected model line is matched to an image line if the image line is no more than 10 pixels from the model line, and if the orientations of the lines differ by no more than $\frac{\pi}{10}$. A hypothesis is accepted if it accounts for at least 50% of the model's lines.

We have performed two sets of experiments. In one, we build models and recognize them, as described above. In the second, we perform indexing using randomly generated models and images. This provides a means of more carefully measuring the discriminatory power of our system.

Figure 3 shows the edges found in two pictures of a telephone, and 15 of the corners that are located from these edges. The corners are numbered for reference. We form groups containing the following sets of corners: ((0 1 2 3 4) (1 2 3 13) (1 0 9 10) (14 15 16 18) (11 9 10 17) (12 13 3 4) (11 17 19))), and for verification we use line segments with the following corners as end points: ((0 1) (1 2) (2 3) (3 4) (0 9) (9 10) (9 11) (11 17) (10 17) (3 13) (13 12) (12 4) (14 15) (15 16) (16 18) (18 14))). Out of these primitive groups, we formed pairs of model groups for entry in the index table, as described above.

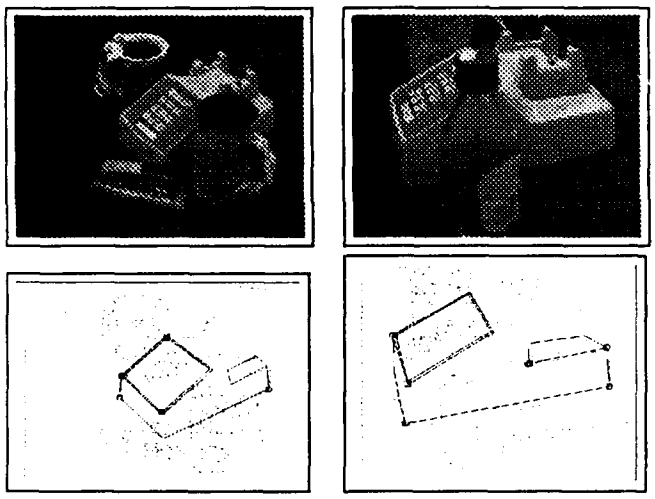


Figure 4: On the top are two scenes containing the phone. Underneath is the first hypothetical projection of the model considered by the recognition system. Both are correct. In the hypotheses, edges are shown as dotted lines, projected line segments appear as lines, circles represent the image corners in the match, and squares show the location of the projected model corners.

There were an average of 472 table entries for each pair of groups.

Figure 4 shows pictures containing the telephone, and the first hypothesis of the recognition system as to the location of the phone. For both pictures, the first hypothesis is correctly accepted. In all, the system was tested on three scenes. Fourteen groups of corners were formed from these images. Indexing for two of the groups required going outside the bounds of the lookup table, so these groups were ignored. Indexing produced a correct match in for all of the remaining twelve groups. Five groups contained seven corners each, and seven groups contained six corners each. The groups with seven corners produced a total of two incorrect matches in addition to the correct matches. Each group could potentially be matched to any of 1,386 groups represented in the lookup table that contained seven points. So indexing reduced the number of incorrect matches by a factor of 3,465. The groups with six corners produced an average of 16.7 incorrect matches, compared to the 2,931 groups in the table they could match. This implies an average speedup of a factor of 244. We can see that indexing can produce tremendous savings in time in this domain, particularly as larger groups are formed, and when we may choose among several groups, using the most distinctive group first.

For the synthetic experiments, we generated random model groups by selecting points at random in a cube, and we generated random image groups by selecting points in a square, in which the minimum and maximum distance between the points differed by no more than a factor of 10. We then matched different orderings of these groups using our indexing system, using the same parameters as above. This allowed us to estimate the likelihood that an image group will match a model group by chance, as we vary the size of the group, and the amount of the error. For each set of values we generated 85 model groups, and for each model group we generated 100 image groups. The results are summarized in figure 5. For error of five pixels, these results are similar to our results with real images. They indicate how dramatically speedups increase with group size.

Number of Points	Error	Images out of Bounds	Speedup
6	5	.7	170
7	5	.4	7,300
8	5	.01	127,000 <
6	7	1.1	90
7	7	.9	850
8	7	.04	127,000 <
6	9	1.44	49
7	9	.9	340
8	9	.02	7,500

Figure 5: Experiments with randomly generated models and images. Points are divided into two groups, and all orderings of these groups are considered. Error is in number of pixels. We give the percentage of images for which error would require them to access the index table outside its boundaries. These images were ignored. Speedup indicates the ratio of total possible matches to matches produced. "X <" indicates that no matches were found, out of X possible matches.

6 Previous Work

Our work is based on a general approach to recognition that links grouping and indexing, an approach developed by [15], and later taken in: [11], [17], [10], [21], [4], and [5].

A number of recent indexing systems for recognizing planar objects, or restricted sets of 3D objects, from arbitrary 3D views have been based on invariant functions of the image [22], [13], [14], [7], [8], [20]. For example, as we have noted, the affine coordinates of a planar model are invariant under affine transformations. [13],[14] use this invariance in their system. Invariants allow these systems to represent the image groups that a model group may produce using a single point in index space.

However, as previously mentioned, [3], [4], and [16] have recently shown that there are no non-trivial invariants when models may consist of arbitrary collections of point features. [16] includes further, related results.

In [4] it is proven that in this domain each model must be represented by a 2D surface if a single lookup table is used. In that paper, and in [18], [14], and [2], lookup tables are constructed that sample this 2D surface by sampling the 2D viewing sphere. These systems suffered from two problems. First, the use of sampling, instead of using analytic methods raises some difficulties. In [4] we found that to perform sufficient sampling required excessive computation both at compile time and at run time. And sampling can result in missing correct matches ([2] discusses a way of bounding this error). Second, it requires a good deal of space to repre-

sent these 2D surfaces. For example, [4] required over 5,000 table entries to represent a model group of five points, and [18] required 2,500 entries to represent pairs of vertices. Our current system requires about 472 table entries for groups containing mostly six or seven points. This comparison may be misleading however. For one thing, space constraints required the system described in [4] to discretize the index space less finely than one would like. Each dimension of that table was divided into 40 parts, compared to 100 parts in the current system. Also in general, space requirements grow as group size grows. Finally, building the lookup table with sampling undoubtedly resulted in many table entries being missed.

Our system is also related to the linear combinations work of [19]. In this paper, a method of projection equivalent to ours is used, and it is shown that any image of an object is a linear combination of two independent images of that object, which is also easily seen from our result. The focus of [19] is on using this result to determine the projection of the model from a new viewpoint, in a manner similar to that described here. However, the results of [19] are not directly useful for an indexing system, because the high dimensionality of the linear subspace described would require excessive storage space in an index table.

7 Conclusions

We have presented a general method for representing 3D models of point features in an indexing table in order to quickly match them to 2D images. In terms of the dimensionality of the space required, this method is optimal. We also present analytic methods of building and accessing the table which ensure that all correct matches are found. Together, we use these results to build a fast and robust recognition system.

In addition, we feel that this work is valuable because it reduces recognition to an extremely simple geometric problem. We show that at its core, the problem of recognizing an object in the absence of error is equivalent to determining which lines a point falls on. When error is present, recognition is equivalent to finding which lines intersect a volume. We use this view of recognition to produce a novel analysis of non-accidental properties, and to answer some outstanding questions about invariants.

References

[1] Biederman, I., 1985. "Human Image Understanding: Recent Research and a Theory". CVGIP

[2] Breuel, T., 1990. "Indexing for Visual Recognition from a Large Model Base." MIT AI Memo 1108.

[3] Burns, J., Weiss, R., and Riseman, E., 1990. "View Variation of Point Set and Line Segment Features." *DARPA IU Workshop*:650-659.

[4] Clemens, D. and Jacobs, D., 1991. "Space and Time Bounds on Model Indexing." *IEEE Trans. PAMI*,13(10):1007-1018.

[5] Clemens, D., 1991. *Region-Based Feature Interpretation for Recognizing 3D Models in 2D Images.* MIT AI TR–1307.

[6] Cutting, J., 1936. *Perception with an Eye for Motion.* Cambridge, MA, USA: MIT Press.

[7] Cyganski, D. and Orr, J., 1985. "Applications of Tensor Theory to Object Recognition and Orientation Determination." *IEEE Trans. PAMI*, 7(6):662-673.

[8] Forsythe, D., Mundy, J., Zisserman, A., and Brown, C., 1990. "Invariance: A New Framework for Vision." *IEEE Proc. ICCV*:598-605.

[9] Grimson, W., Huttenlocher, D., and Jacobs, D., 1991. "Affine Matching with Bounded Sensor Error: A Study of Geometric Hashing and Alignment." MIT AI Memo 1250.

[10] Huttenlocher, D. and Wayner, P., 1991. "Finding Convex Edge Groupings in an Image." *IEEE Proc. CVPR*:406-412.

[11] Jacobs, D., 1989. "Grouping for Recognition." MIT AI Memo 1177.

[12] Jacobs, D., 1991. "Optimal Matching of Planar Models in 3D Scenes." *IEEE Proc. CVPR*:269-274.

[13] Lamdan, Y., Schwartz, J. and Wolfson, H., 1988. "Object Recognition by Affine Invariant Matching." *IEEE Proc. CVPR*:335-344.

[14] Lamdan, Y. and Wolfson, H., 1988. "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme." *IEEE Proc. Robotics and Automation*:238-249.

[15] Lowe, D., 1985. *Perceptual Organization and Visual Recognition.* The Netherlands: Kluwer Academic Publishers.

[16] Moses, Y. and Ullman, S., 1991. "Limitations of Non Model-Based Recognition Schemes." MIT AI Memo 1301.

[17] Sha'ashua, A. and Ullman, S., 1988. "Structural Saliency: The Detection of Globally Salient Structures Using a Locally Connected Network." *IEEE ICCV*:321-327.

[18] Thompson, D. and Mundy, J., 1987. "Three-Dimensional Model Matching from an Unconstrained Viewpoint." *IEEE Proc. Robotics and Automation*:208-220.

[19] Ullman, S. and Basri, R., 1991. "Recognition by Linear Combinations of Models." *IEEE Trans. PAMI*,13(10):992-1007.

[20] Van Gool, L., Kempenaers, P., and Oosterlinck, A., 1991. "Recognition and Semi-Differential Invariants." *IEEE Proc. CVPR*:454-460.

[21] Wayner, P., 1991. "Efficiently Using Invariant Theory for Model-based Matching." *IEEE Proc. CVPR*:473-478.

[22] Weiss, I., 1988. "Projective Invariants of Shape." *DARPA IU Workshop*:1125-1134.

3-D Model Alignment Without Computing Pose

Joseph L. Mundy and **Richard P. Welty**
General Electric CR&D
Schenectady, New York

Michael H. Brill
Science Applications International Corporation
McLean, Virginia

Paul M. Payton and **Eamon B. Barrett**
Lockheed Missiles and Space Company
Sunnyvale, California

ABSTRACT

Most current object recognition systems are based on a 3D model which is used to describe the image projection of an object over all viewpoints. In this paper we introduce a new technique which can predict the geometry of an object under projective transformation. The object geometry is represented by a set of corresponding features taken from two views. The projected geometry can be constructed in any third view, using a viewpoint invariant derived from the correspondences.

1 Introduction

The central focus of object recognition research over the last decade has been on the use of 3D geometric models as a representation of objects. These geometric models are assumed to characterize the significant features of the object. The main idea is that the model geometry is independent of viewpoint and that the model can be used to predict features easily extracted from the image. In current practice, the model and image features are represented by points and lines. The image is initially processed to extract a 2D geometric description of the entire image plane. Then conceptually, the 3D model is projected over all viewpoints and the projected model geometry is compared with the geometric features extracted from the image.

In order to recognize a number of classes of objects it is necessary to develop a 3D geometric model for each one. In some systems, the modeling is done directly with a standard CAD modeling package. In other systems it is possible to acquire model representations, with some manual interaction, directly from multiple image views of the object. These approaches are quite time consuming and it is often difficult to construct a valid three dimensional model so that the topological connections between faces, edges and vertices are all correct.

In this paper we focus on the issue of model construction. Instead of constructing a full 3D representation of the object, we develop a method which exploits correspondences between two views of the object. As we will show, it is sufficient to determine eight corresponding points between two model images of an object and then the projection of the object in a new third view can be determined from eight additional correspondences. The technique is based on the general notion of viewpoint invariance [2, 3, 5] where geometric properties of an object are derived which are invariant to projective transformation.

This approach is similar to the work of Koenderink [6] who has described a technique for transferring the shape of a three dimensional configuration of points under an affine transformation. He uses two views of the points to

define the 3D affine coordinates of the points. The correspondence between four points in two images is sufficient to determine an affine reference frame for all of the remaining points. Once such a frame is established, it is then possible to transfer the projection of the set of points to any new viewpoint by determining four new corresponding points. Our techniques use the linear fractional or perspective transformation, rather than the affine transformation, in order to model the camera geometry more accurately.

The central idea is that one can construct a pair of 2D models for an object (one from each of the model views) and then use these models for registration and recognition in any third view of the object[1] We develop the idea by starting with projections in one dimension and proceeding to the two and three dimensional cases.

2 Model Transfer

2.1 Projection on a Line

Consider a line with three given points, (p_1, p_2, p_3) and a general point, x. It is a well known result in projective geometry that the cross ratio, Cr is preserved under projective mappings of the line. For example, in Figure 1 the points are mapped by a central projection onto another line. The projected points are denoted, $(\overline{p_1}, \overline{p_2}, \overline{p_3})$ and $\overline{x}$. The cross ratio of the four points on both lines is a scalar invariant, i.e.,

$$Cr = \frac{(x - p_1)(p_2 - p_3)}{(x - p_2)(p_1 - p_3)} = \frac{(\overline{x} - \overline{p}_1)(\overline{p}_2 - \overline{p}_3)}{(\overline{x} - \overline{p}_2)(\overline{p}_1 - \overline{p}_3)}. \quad (1)$$

This
equation relates the postion of point, $\overline{x}$ with the position of point, x, in terms of of the point correspondences, $[(p_1, \overline{p_1}), (p_2, \overline{p_2}), (p_3, \overline{p_3})]$. Equation 1 can be rewritten as a bilinear transformation relating x and $\overline{x}$ as follows,

$$\overline{x} = \frac{\alpha x + \beta}{\gamma x + \delta} \quad (2)$$

Thus three correspondences define a mapping between two lines as a projective transformation. Once the correspondences are known, any number of points x_i can be transformed by equation 2.

[1]There is no notion of occlusion in this process, so that more than two model views will be required in general to capture all of the features of the object.

2.2 Model Transfer in Two Dimensions

The same process as just described for transforming points on a projective line can be extended to the case of mapping between projective planes. When the points are represented in homogeneous coordinates an arbitrary projective transformation between planes is a linear form, $\overline{\mathbf{X}} = \mathsf{T}\mathbf{X}$, where $X = (x, y, 1)^t$, and T is a 3x3 matrix. We can define an invariant similar to the cross ratio on the line as follows.

Let (p, q) be the point whose referent $(\overline{p}, \overline{q})$ in the new image is to be determined. The cross-ratio of determinants is known to be a projective invariant. Therefore:

$$\frac{\begin{Vmatrix} p_1 & p_2 & p_3 \\ q_1 & q_2 & q_3 \\ 1 & 1 & 1 \end{Vmatrix} \begin{Vmatrix} p_1 & p_4 & p \\ q_1 & q_4 & q \\ 1 & 1 & 1 \end{Vmatrix}}{\begin{Vmatrix} p_1 & p_2 & p_4 \\ q_1 & q_2 & q_4 \\ 1 & 1 & 1 \end{Vmatrix} \begin{Vmatrix} p_1 & p_3 & p \\ q_1 & q_3 & q \\ 1 & 1 & 1 \end{Vmatrix}} \triangleq \mathsf{C}(1, 2, 3, 4, p, q) \quad (3)$$

$$\frac{\begin{Vmatrix} \overline{p}_1 & \overline{p}_2 & \overline{p}_3 \\ \overline{q}_1 & \overline{q}_2 & \overline{q}_3 \\ 1 & 1 & 1 \end{Vmatrix} \begin{Vmatrix} \overline{p}_1 & \overline{p}_4 & \overline{p} \\ \overline{q}_1 & \overline{q}_4 & \overline{q} \\ 1 & 1 & 1 \end{Vmatrix}}{\begin{Vmatrix} \overline{p}_1 & \overline{p}_2 & \overline{p}_4 \\ \overline{q}_1 & \overline{q}_2 & \overline{q}_4 \\ 1 & 1 & 1 \end{Vmatrix} \begin{Vmatrix} \overline{p}_1 & \overline{p}_3 & \overline{p} \\ \overline{q}_1 & \overline{q}_3 & \overline{q} \\ 1 & 1 & 1 \end{Vmatrix}} \triangleq \overline{\mathsf{C}}(1, 2, 3, 4, \overline{p}, \overline{q}). \quad (4)$$

The invariant condition that

$$\mathsf{C}(1, 2, 3, 4, p, q) = \overline{\mathsf{C}}(1, 2, 3, 4, \overline{p}, \overline{q}) \quad (5)$$

determines a linear relationship between the unknown quantities $(\overline{p}, \overline{q})$ and all the other variables which are known.

By interchanging points 1 and 2 in this procedure, a second equation is generated:

$$\mathsf{C}(2, 1, 3, 4, p, q) = \overline{\mathsf{C}}(2, 1, 3, 4, \overline{p}, \overline{q}). \quad (6)$$

The intersection of these lines determines the point $(\overline{p}, \overline{q})$, as illustrated in Figure 2. The quantities $\mathsf{C}(1, 2, 3, 4, p, q)$ and $\mathsf{C}(2, 1, 3, 4, p, q)$ are projective coordinates (with respect to the reference points $1, 2, 3, 4$) for the point (p, q). This transfer technique exploits the invariance of these projective coordinates.

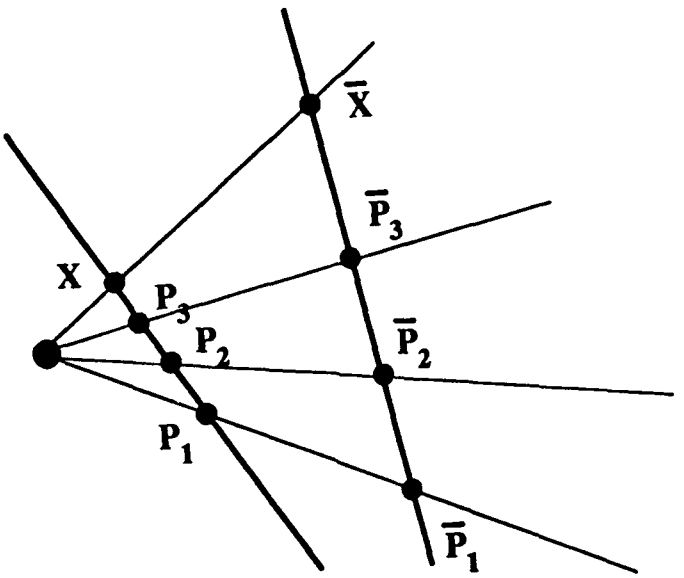


Figure 1: The projection of one line onto another. Four points define an invariant cross ratio.

Note that this method requires neither knowledge of the reference objects (x_j, y_j), $j = 1 \ldots 4$, nor of the imaging system parameters. The transformation is defined by four accurately determined correspondences between reference image points (p_j, q_j) $j = 1 \ldots 4$, and their corresponding points $(\overline{p}_j, \overline{q}_j)$ $j = 1 \ldots 4$ in the new image.

3 Model Transfer From Two Views in Three Dimensions

Suppose we have two reference images and wish to transfer the corresponding reference image point pairs: (p, q), $(\overline{p}, \overline{q})$, to a new image point (w, s). The camera models for the two images are not given nor are there ground control points available to estimate these models. Model transfer can still be accomplished by a method involving projective invariants derived for two views with eight correspondences: (p_j, q_j), $(\overline{p}_j, \overline{q}_j)$ $j = 1 \ldots 8$, between the two reference images, and (w_j, s_j) $j = 1 \ldots 8$, on a new third image. The following development is a natural extension of the eight point camera motion result of Longuet-Higgens [7]. He demonstrated that the relative motion between two views of a calibrated camera can be determined by linear methods from eight point correspondences. Thus any ninth point can be projected from the known camera motion. The result here is a significant extension in that **no** camera calibration is required.

The projection equations of two perspective cameras can be manipulated to the following form. In the case of two images of the same point, the general form of these equations is:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \overline{a}_{11} & \overline{a}_{12} & \overline{a}_{13} & \overline{a}_{14} \\ \overline{a}_{21} & \overline{a}_{22} & \overline{a}_{23} & \overline{a}_{24} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} p & & & \\ & q & & \\ & & \overline{p} & \\ & & & \overline{q} \end{pmatrix} \begin{pmatrix} b_1 & b_2 & b_3 & b_4 \\ b_1 & b_2 & b_3 & b_4 \\ \overline{b}_1 & \overline{b}_2 & \overline{b}_3 & \overline{b}_4 \\ \overline{b}_1 & \overline{b}_2 & \overline{b}_3 & \overline{b}_4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (7)$$

or

$$\left[\mathsf{A} - \begin{pmatrix} p & & & \\ & q & & \\ & & \overline{p} & \\ & & & \overline{q} \end{pmatrix} \mathsf{B} \right] \mathbf{x} = 0. \quad (8)$$

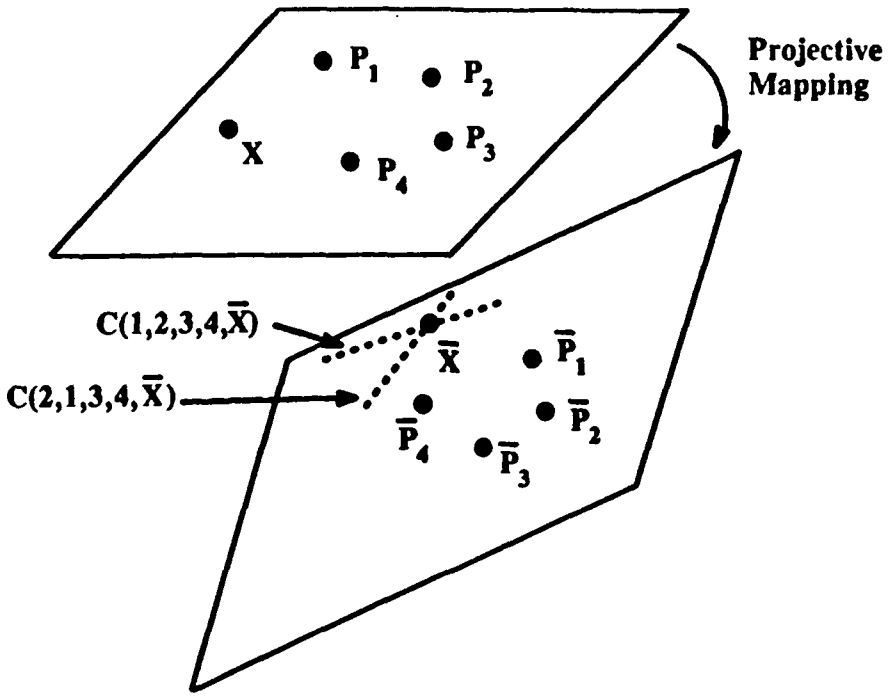


Figure 2: Transfer of a point in two dimensions using the five-point invariants. The two invariants define a pair of linear constraints which define the point.

From the latter equation we infer that:

$$\left\| \mathsf{A} - \begin{pmatrix} p & & & \\ & q & & \\ & & \overline{p} & \\ & & & \overline{q} \end{pmatrix} \mathsf{B} \right\| = 0. \qquad (9)$$

Note that A and B contain parameters associated with the imaging geometry of the two cameras and (p, q), $(\overline{p}, \overline{q})$ are corresponding points in the left and right images. We observe that, expanding the determinant in terms of the variables $(p, q, \overline{p}, \overline{q})$ we have:

$$0 = \alpha_1 pq\overline{p}\,\overline{q} + \alpha_2 pq\overline{p} + \cdots + \alpha_{14}\overline{p} + \alpha_{15}\overline{q} + \alpha_{16}. \qquad (10)$$

Examining the minor determinants, we observe that $\alpha_1 \ldots \alpha_7 = 0$ since they involve repeated instances of the same row-vector (in α_1 for example, the row-vector b is repeated twice, as is $\overline{\mathsf{b}}$.)

Thus the vectors:

$$(\alpha_8\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\alpha_{13}\alpha_{14}\alpha_{15}\alpha_{16})$$

and

$$(p_j\overline{p}_j,\ p_j\overline{q}_j,\ q_j\overline{p}_j,\ q_j\overline{q}_j,\ p_j,\ q_j,\ \overline{p}_j,\ \overline{q}_j, 1)$$

are orthogonal for every j. Similarly a nine-component vector $\vec{\beta}$, orthogonal to the vector:

$$(p_j w_j,\ p_j s_j,\ q_j w_j,\ q_j s_j,\ p_j,\ q_j,\ w_j,\ s_j,\ 1)$$

and a third nine-component vector $\vec{\gamma}$, orthogonal to the vector:

$$(\overline{p}_j w_j,\ \overline{p}_j s_j,\ \overline{q}_j w_j,\ \overline{q}_j s_j,\ \overline{p}_j,\ \overline{q}_j,\ w_j,\ s_j,\ 1)$$

can be found. The components of these vectors may be determined from the relations, given eight corresponding points in each image:

$$\begin{vmatrix} p_1\overline{p}_1 & \cdots & p_8\overline{p}_8 & \alpha_8 \\ p_1q_1 & \cdots & p_8q_8 & \alpha_9 \\ q_1\overline{p}_1 & \cdots & q_8\overline{p} & \alpha_{10} \\ q_1\overline{q}_1 & \cdots & q_8\overline{q}_8 & \alpha_{11} \\ p_1 & \cdots & p_8 & \alpha_{12} \\ q_1 & \cdots & q_8 & \alpha_{13} \\ \overline{p}_1 & \cdots & \overline{p}_8 & \alpha_{14} \\ \overline{q}_1 & \cdots & \overline{q}_8 & \alpha_{15} \\ 1 & \cdots & 1 & \alpha_{16} \end{vmatrix} = 0$$

and similar relations for $\beta_8 \ldots \beta_{16}$ and $\gamma_8 \ldots \gamma_{16}$ are determined from combinations of (p_j, q_j) with (w_j, s_j), and $(\overline{p}_j, \overline{q}_j)$ with (w_j, s_j), respectively.

The $\vec{\alpha}, \vec{\beta}$ and $\vec{\gamma}$ vectors are readily seen to be proportional to minors of these determinants.

Once $\vec{\beta}$ and $\vec{\gamma}$ have been determined, we observe that transfer can be accomplished, as follows: Suppose that a corresponding point pair (p,q) and $(\overline{p},\overline{q})$ has been selected from the reference images. We wish to determine the point (w,s) in the new image corresponding to these reference points. We observe that the relation:

$$0 = \beta_8 pw + \beta_9 ps + \beta_{10} qw + \beta_{11} qs + \beta_{12} p + \beta_{13} q + \beta_{14} w + \beta_{15} s + \beta_{16} \quad (11)$$

when evaluated, equals:

$$\mathsf{A}^{(1)} w + \mathsf{B}^{(1)} s + \mathsf{C}^{(1)} = 0, \quad (12)$$

i.e. the (w,s) solution lies on the line in the (w,s) image plane.

Similarly, using the second reference image we observe the second relation:

$$0 = \gamma_8 \overline{p} w + \gamma_9 \overline{p} s + \gamma_{10} \overline{q} w + \gamma_{11} \overline{q} s + \gamma_{12} \overline{p} + \gamma_{13} \overline{q} + \gamma_{14} w + \gamma_{15} s + \gamma_{16} \quad (13)$$

determines a second line:

$$\mathsf{A}^{(2)} w + \mathsf{B}^{(2)} s + \mathsf{C}^{(2)} = 0 \quad (14)$$

in the (w,s) image plane. The intersection of these lines, is the solution.

Note that additional correspondences can be used to define the transfer vectors,α,β,γ. The solution is still linear and formulated in terms of the generalized inverse of a non-square matrix.

4 Experiments

To demonstrate the validity of the new model transfer approach, we have carried out a number of experiments on aerial photographs of military and urban sites. It is emphasized that no ground control points or three dimensional models were available. The model projections are achieved without knowledge of the camera model for any of the images.

4.1 Ideal Data

The first experiment involves an ideal geometric wireframe model. The eight correspondences are selected as the base structure in the upper right and lower left images in Figure 3. The vertices of the small cube are then transferred to the third view at the upper left. The relative positions of the views is similar to two nearly nadir views of a structure resulting in the formation of a synthetic oblique view. The wire frame model is expressed solely in terms of the two dimensional vertex locations in each view.

4.2 Lockheed Site

In this experiment, we demonstrate an important application of model transfer to the problem of change detection. We have two views of a building on the Lockheed site in Sunnyvale, CA. In a third view, taken much earlier, the building was not yet in existence. We use manually selected correspondences from the buildings nearby to generate the transfer vectors. Then points from the footprint of the new building are transferred to the old image. The building location at the time of the original image was an empty field.

In Figure 4 we show the two original images and a few segments of the footprint of the building projected onto the older image. The two outlines illustrate the effect of using 8, 10 and 12 reference points for the transfer. The results for 10 and 12 points lie practically on top of each other. In most of the experiments, there is a noticeable improvement in projection accuracy for 8 and 10 points, but there is little further change in going from 10 to 12 reference points. Again, in this experiment, the reference points were selected manually.

4.3 Segmentation-Guided Transfer

We then investigated the use of automatic segmentation techniques to derive the position of reference points in each image. These experiments were carried out on views of the Schenectady 109th Tactical Airlift Base.

The images were segmented using a modified form of the Canny [4] edge detector. The modified version carries out a more thorough analysis of the intensity near corners than the original algorithm. Vertices are derived by breaking the boundary edge chains at points of high local curvature [1]. An example of the segmentation is shown in Figure 5. The location of these vertices is used as the position of reference points in the model transfer calculations.

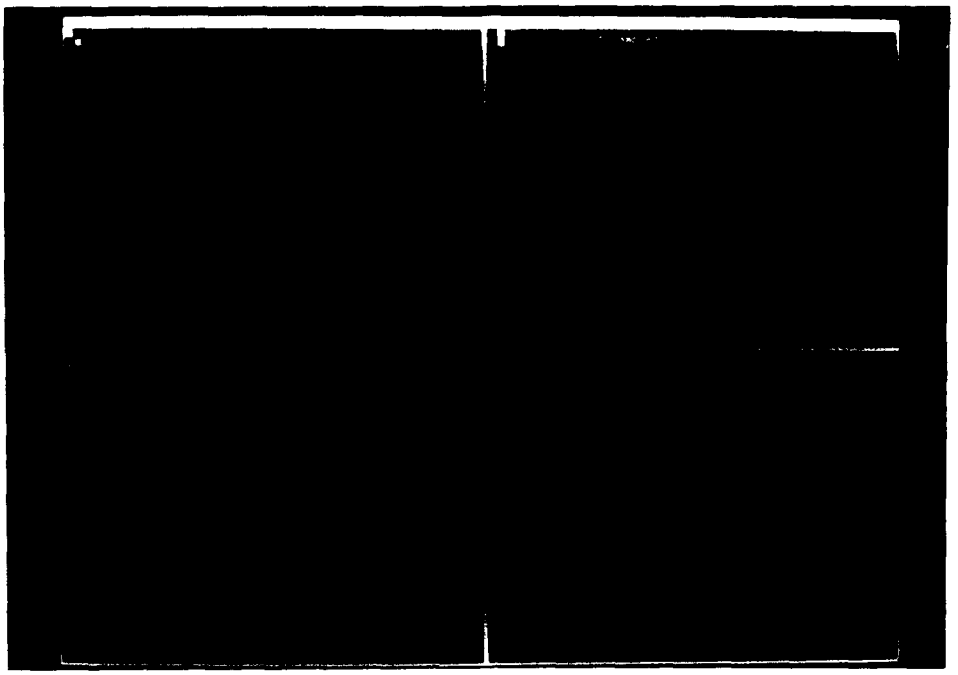


Figure 3: Model transfer on ideal data. The image at lower left is a oblique view of the small cube which illustrates the model transfer process.

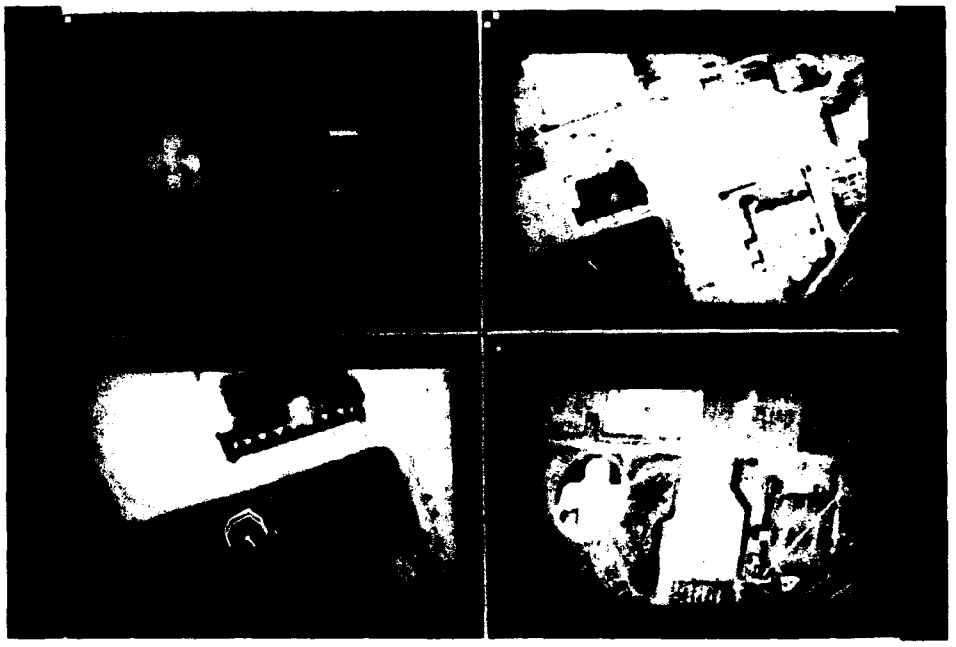


Figure 4: Two recent views of the Lockheed site are used to define the location of a building in an earlier view prior to the building construction. The lower left view illustrates the transfer for 8, 10 and 12 reference points; the upper right view is zoomed out to provide more context.

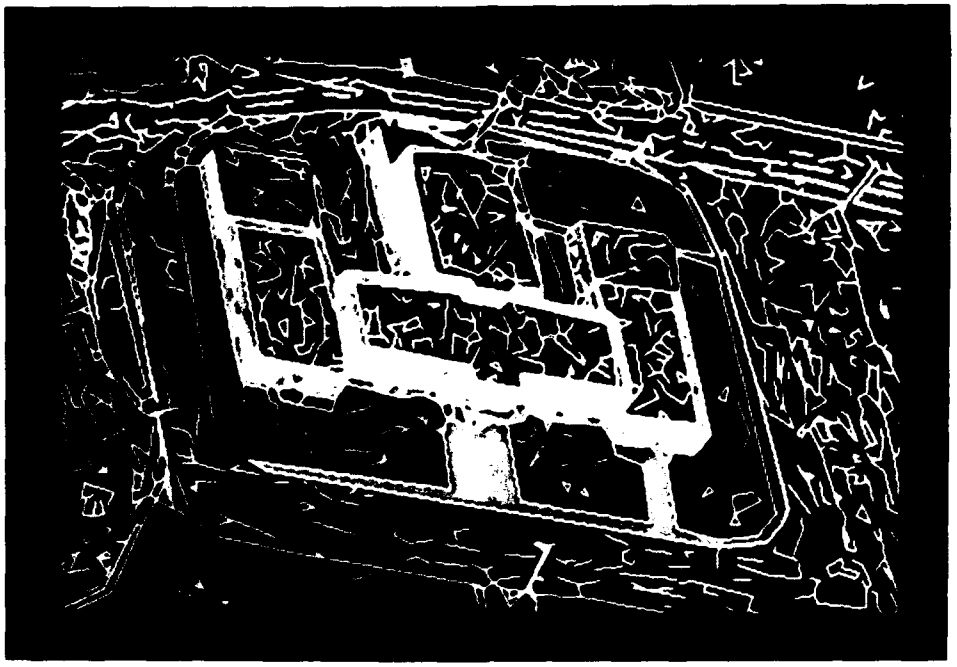


Figure 5: The segmentation of intensity boundaries in views of the Schenectady Air Base. The segmentation is the result of a modified Canny edge detector followed by corner detection, based on curvature.

In this experiment, the goal is to transfer an edge of the main command building from two given model views into a third, quite oblique view. In an actual application, a complete model of the building can be constructed and transferred. Here we focus on one edge of the building to illustrate the accuracy of the method.

In Figure 6 we illustrate the worst case performance of the segmentation generated vertices. In this view, the edge of the building was in a shadow, so one boundary was not recovered by the segmentation algorithm. Instead, the vertex was associated with a nearby feature. The resulting trihedral junction for the corner of the building is shown in the top right image.

Even with this error, the model transfer was achieved with quite reasonable accuracy. In Figure 7 the transfer of one edge of the building is shown for 8, 10 and 12 reference points. Again there is a significant change from 8 to 10 points with little change in going to 12 points.

5 Conclusions

These experiments demonstrate the practical feasibility of the proposed model transfer technique. The segmentation experiments were quite encouraging and we have demonstrated that it will be feasible to automate the process. There are a number of obvious extensions which we are now pursuing.

- An initial model transfer is used to guide the segmentation of additional features. In this way we could have used a more sensitive edge extraction threshold in the shadow region of the building in the last example.
- It is also possible to derive an equivalent form of the transfer relation for lines instead of points. Given that line segments are more accurately and consistently extracted by automatic segmentation techniques, the performance of the technique should be improved.
- Use the invariant relation to automatically locate corresponding features. It is not nec-

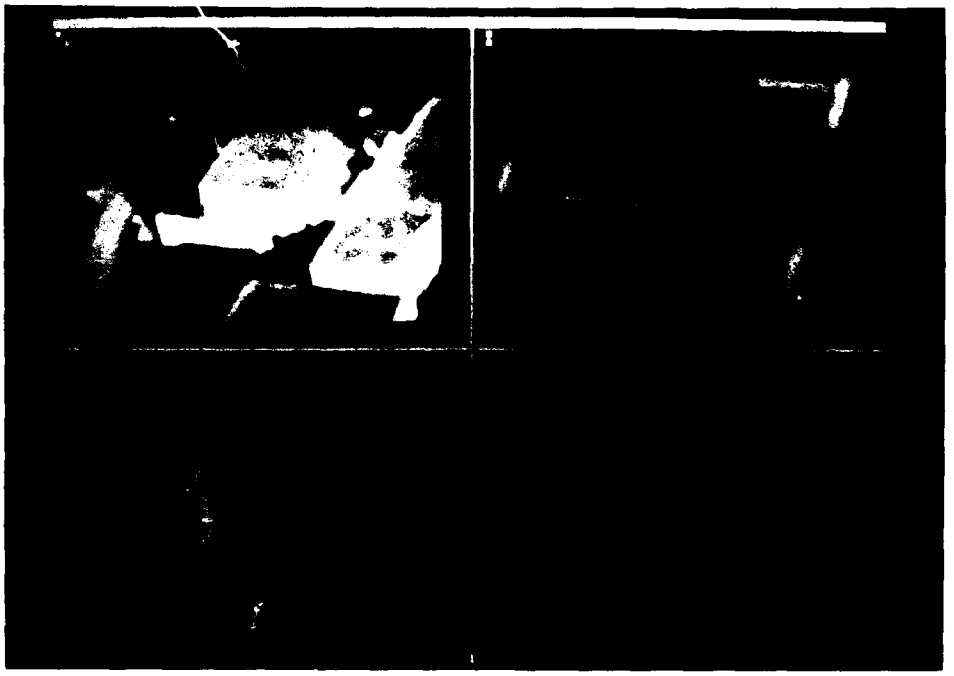


Figure 6: The corner of a building is located using the vertices defined by automatic segmentation. Note that in the upper right image, one edge of the building is in shadow and the associated vertex locations are not correctly extracted.

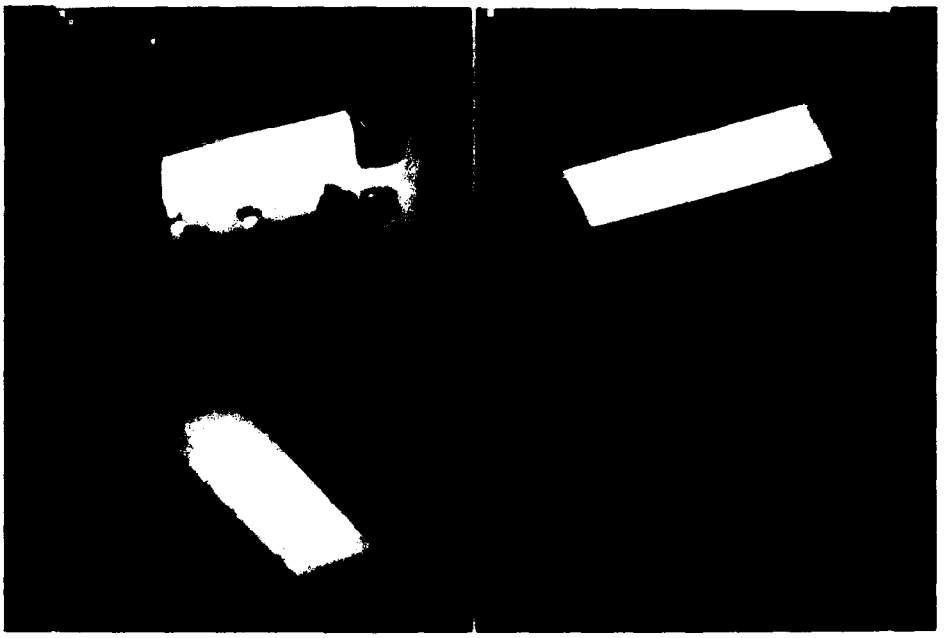


Figure 7: The transfer based on segmentation derived features. The edge of the motor pool is transferred from two reference views into a third. Each pair of images is used to map onto the remaining image. The results are shown for 8, 10, and 12 reference points.

essary to consider the features as isolated, unrelated points. In many cases, the footprint of a building or other structure can be extracted in a partially connected topology which provides some grouping of the features to reduce the combinatorial cost. The consistency of feature correspondences can be determined by evaluating the invariant conditions.

References

[1] H. Asada and M. Brady. The curvature primal sketch. In *Proc. IEEE Workshop on Computer Vision: Representation and Control*, 1984.

[2] E. B. Barrett, M. H. Brill, N. N. Haag, a. P. M. Payton. Invariant linear methods in photogrammetry and model-matching. In J. L. Mundy and A. Zisserman, editors, *Applications of Invariance in Computer Vision*. MIT Press, 1992. To Appear.

[3] E. B. Barrett, P. M. Payton, N. N. Haag, and M. H. Brill. General methods for determining projective invariants in imagery. *Computer Vision, Graphics, and Image Processing*, 53(1):46–65, Jan. 1991.

[4] J. Canny. Finding edges and lines in images. Report AI-TR-720, MIT, Artificial Intelligence Laboratory, 1983.

[5] D. A. Forsyth, J. L. Mundy, A. P. Zisserman, and C. M. Brown. Projectively invariant representations using implicit algebraic curves. In *Proc. First European Conf. Comput. Vision*, LNCS. Springer, 1990.

[6] J. J. Koenderink and A. J. Van Doorn. Affine structure from motion. *Journal of the Optical Society of America*, A, Vol 8.(2), February 1991.

[7] H. C. Longuet-Higgens. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–235, 1981.

APPLICATION OF POSE DETERMINATION TECHNIQUES TO MODEL EXTENSION AND REFINEMENT

Rakesh Kumar and Allen R. Hanson

Computer and Information Science Department
University of Massachusetts at Amherst *

Visual measurements of modelled 3D landmarks provide strong constraints on the location and orientation of a mobile robot. To make the landmark-based robot navigation approach widely applicable, it is necessary to be able to automatically build the landmark models. A substantial amount of effort has been invested by computer vision researchers over the past ten years on developing robust methods for computing 3D structure from a sequence of 2D images. However, robust computation of 3D structure, with respect to even small amounts of input image noise, has remained an open problem. The approach adopted in this paper is one of model extension and refinement. A partial model of the environment is assumed to exist and this model is extended over a sequence of frames. As will be shown in the experiments, the prior knowledge of the small partial model greatly enhances the robustness of the 3D structure computations. The initial 3D model may have errors and these are also refined over the sequence of frames.

1 INTRODUCTION

An important problem in vision is to automatically build 3D models of objects and scenes. In a previous paper [10], least-squares and robust methods were presented for determining the location and orientation of a robot from visual measurements of modeled 3D landmarks. However, building the 3D landmark models is a time consuming and tedious affair. For the landmark-based navigation methods to be widely applicable, automatic methods have to be developed to build and enhance the 3D models. Ideally, the robot would continuously build and update its world model as it explores the environment. This paper presents techniques to determine the 3D location of image features from a sequence of 2D image frames taken by a camera mounted on the robot. It is assumed that a prior partial model is available. The goal is to have the robot extend and refine this model as it explores the world.

Extensive research has been done in computer vision to develop robust algorithms for extracting 3D information from a sequence of 2D images. Of the many different visual cues for extracting 3D information, the two most extensively researched are stereo and motion. The basic principle exploited in both cues is triangulation (see Figure 1). New points are located by triangulating the projection rays from corresponding points in two or more frames.

In applications involving stereo, two cameras separated by a baseline are used to do the triangulation. The two cameras are fixed with respect to each other and therefore the relative orientation is determined during a prior calibration stage. Thus, the main problem and focus of stereo research has been to establish correspondences [12].

In two-frame motion analysis both the correspondences and the relative orientation between the two camera frames are unknown. Research in motion analysis has classically been divided into two steps. In the first step inter-frame image displacements of image pixels and/or higher level tokens are computed. The second step, also known as "Structure from Motion" or "Relative Orientation", is the interpretation of these displacements (or correspondences between image tokens) into 3D structure and relative orientation (rotation and translation) between frames [1, 9].

However, due to noise in the measurement process, results for both stereo and motion analysis from using just two frames are not very robust [1, 8]. To improve the robustness of the results, the traditional stereo and structure from motion techniques have been extended to deal with multi-frame image sequences [3, 5, 13, 14, 16], under the assumption that temporal integration would lead to more robust results.

The multi-frame research can be categorised into two broad classes or strategies. The first class assumes that a model of 3D inter-frame motion is known, rather than assuming independent motion parameters between consecutive frames. Broida [5] assumes constant velocity motion and estimates the 3D location of a set of points tracked over a monocular image sequence. Recently, Chandrasekhar et. al. [6] have extended Broida's technique to deal with data sets where the 3D location of a few points is known. The objective function, which Broida and Chandrasekhar et. al. minimize has the motion model parameters and the unknown structure locations as unknowns. Thus the dimension of the objective function grows with the number of unknown points. An

*This research was supported by the following Defense Advanced Research Projects Agency grants DAAE07-91-C-R035, DACA76-89-C-0017 and National Science Foundation grant CDA-8922572.

even more basic limitation of this approach lies in the model of motion being adopted and its suitability to the motion being observed.

The second class of techniques does not assume any model of motion. The rigid structure of the world is carried forward by the depth estimates from frame to frame. These techniques are sequential in nature and typically use Kalman Filtering to compute the depth estimates[3, 7, 13, 14, 16].

Both, Ayache et. al. [3] and Zhang et. al. [16] build world models using multi-frame stereo sequences. Zhang et. al. [16] track 3D line segments over a sequence of stereo image frames and use a Kalman Filter to integrate the results for a final 3D estimate of the 3D line segment. To do the temporal integration, the absolute orientation between successive stereo-pair coordinate frames is determined.

Oliensis and Thomas [13] use Horn's relative orientation algorithm [9] to solve for the motion parameters between consecutive image frames in a monocular image sequence. With each image pair, new measurements are made for depth values of features and these are integrated with previous estimates in the Kalman Filter framework. The new observation Oliensis and Thomas [13] make is that the depth estimate of different feature points are correlated since the same noisy motion parameters are used to compute the depth. Because of this correlation, they estimate the depth parameters of all points simultaneously. This gives them fairly good depth estimates for camera motions having some T_z (i.e. translation along the optical axis) component. The cost, however, is that for estimating the depths of "m" points, a covariance matrix of size (3m x 3m) must be inverted with each new frame.

Sawhney et. al. [14] also use Kalman Filtering to estimate the depths of "shallow structures" over a monocular sequence of multiple image frames (shallow structures are those whose extent in depth is small compared to their average depth from the camera). The algorithm, however, cannot handle non-shallow structures. The image motion of shallow structures can be described by an affine transform. Based on the affine trackability of an object, they are able to segment out different shallow structures in the scene and hence can potentially handle multiple moving objects. In an experiment reported in the results section of this paper, an initial model is built using the 3D points lying on some of the shallow structures recovered by their algorithm. Using this initial model, the 3D location of other points in the scene is estimated by the techniques developed in this paper. Thus with a combination of techniques presented in this paper and Sawhney et. al.'s [14] technique for 3D recovery of shallow structures, a fairly robust general motion technique may be constructed.

1.1 Our Approach

The approach adopted here is to first begin with a partial model (possibly noisy) and to then extend and refine it by viewing the object over a sequence of frames. Both modeled and unmodeled features of the object are tracked over the image sequence by using an optic flow based line tracking algorithm [2, 15]. Correspondences are obtained between the modeled 3D features and their image projections. Using the flow of image tokens and the poses of the object computed from model-image feature correspondences for a sequence of image frames, new points are located by triangulation (see Figure 1). The triangulation process is also used to make new 3D measurements of the initial model points. These measurements are then fused with the previous estimates to refine the set of initial model points. The approach adopted here is basically induced stereo. Tracking image features over a large sequence effectively leads to a large baseline for stereo and improves the robustness of the 3D reconstructions. Note that this approach does not require any models of inter-frame motion.

The key assumption made is that a partial model is available at the beginning of the process. Due to the availability of the partial model, new points are located in a stable world coordinate system. The pose computed for each frame are independent of the other frames, so each frame provides an independent measure to the whole process[1]. This does not lead to the cascading problems which most of the sequential multi-frame "structure from motion" techniques suffer from because noisy prior estimates in the previous frame's coordinate system are integrated with new estimates in the current frame's coordinate system.

The estimation of the new 3D points is done using both batch and quasi-batch or sequential methods. Triangulation requires at least two frames and therefore the minimum batch size is two. Results from batch to batch are integrated by the standard Kalman Filter covariance based updating equations. Results are presented for three real data sequences where new 3D points are located with average errors less than 1.7 % . These results are far superior to those obtained by the traditional structure from motion techniques employed in computer vision. This supports the earlier stated premise that prior knowledge of a partial model greatly extends the robustness of the structure estimates.

The errors in the initial partial model are assumed to be either gross errors or gaussian noise. If gross errors are present in the 3D model, these would be detected as outliers by the robust pose recovery techniques developed in our earlier paper [10] and would not be used for the final step of least-squares fitting to the remaining non-outlier data. Note that outliers can also arise due to incorrect correspondences. However, if a modeled landmark appears as an outlier over a large number of frames, then it probably is due to a gross error in the 3D model and it could eventually be removed from the 3D model database. Thus for the remainder of this paper, the noise in the input 3D model is assumed to be gaussian. Section 2 extends the least-squares algorithms for pose determination (presented in [10]) to handle gaussian noise both in the 3D model and image measurements. Section 3 presents the mathematics for locating new points

[1] Note, this would not be true if there was significant noise in the initial partial model.

and refining old points using the computed poses and their respective variances. Finally, Section 4 presents and analyses results from real data experiments. Some concluding remarks are presented in Section 5.

2 Pose Determination

In an earlier paper [10] least-squares techniques for pose determination were developed. These techniques are optimal with respect to gaussian noise in the input image measurements. In this section, the least-squares techniques are extended to handle gaussian noise in the 3D model. The techniques presented in this section assume point correspondences but are easily modified for line correspondences.

The rigid body transformation from the world coordinate system to the camera coordinate system can be represented as a rotation (R) followed by a translation ($\vec{T}$). The point $\vec{p}$ in world coordinates gets mapped to the point $\vec{p_c}$ in camera coordinates:

$$\vec{p_c} = R(\vec{p}) + \vec{T} \tag{1}$$

Using equation (1) and assuming perspective projection, the pose constraint equations for the i'th point $\vec{p_i}$ in a set of "m" points can be written in the following manner:

$$\frac{1}{p_{czi}}\vec{C}_{xi} \cdot (R\vec{p_i} + \vec{T}) = 0 \tag{2}$$

$$\frac{1}{p_{czi}}\vec{C}_{yi} \cdot (R\vec{p_i} + \vec{T}) = 0 \tag{3}$$

$$\vec{C}_{xi} = (s_x, 0, -I_{xi}) \tag{4}$$

$$\vec{C}_{yi} = (0, s_y, -I_{yi}) \tag{5}$$

$$p_{czi} = (R\vec{p} + \vec{T})_z \tag{6}$$

(I_{xi}, I_{yi}) is the image projection of the point and (s_x, s_y) is the focal length in pixels along each axis.

Since both the image measurements and the 3D model locations are assumed to be noisy, it will not be possible to satisfy the above constraint equations exactly. Let the measurement error in pixels of image point locations be given by $(\Delta X, \Delta Y)$ and the error in the 3D model points be given by $\vec{\Delta p}$. Given a current estimate $R, \vec{T}$, the constraint equations (2,3) are linearised about the estimate:

$$\frac{1}{p_{czi}}(\vec{C}_{xi} \cdot \vec{\Delta T} + \vec{\delta\omega} \cdot \vec{b_i}) = -\frac{1}{p_{czi}}\vec{C}_{xi} \cdot \vec{p}_{ci} + \eta_x \tag{7}$$

$$\frac{1}{p_{czi}}(\vec{C}_{yi} \cdot \vec{\Delta T} + \vec{\delta\omega} \cdot \vec{b_i}) = -\frac{1}{p_{czi}}\vec{C}_{yi} \cdot \vec{p}_{ci} + \eta_y \tag{8}$$

where η_x and η_y are the noise terms in the two equations. η_x and η_y are functions of both model noise $\vec{\Delta p}$ and image noise $\Delta X, \Delta Y$:

$$\eta_x = \Delta X + \frac{1}{p_{czi}}\vec{C}_{xi} \cdot (R(\vec{\Delta p_i})) \tag{9}$$

$$\eta_y = \Delta Y + \frac{1}{p_{czi}}\vec{C}_{yi} \cdot (R(\vec{\Delta p_i})) \tag{10}$$

Therefore for the i'th point, two such equations (7 and 8) can be written and for a set of "m" points, a total of "2m" equations is obtained. This system of "2m" equations is similar to the linear system of equations (18) described in the Appendix. This linear system of equations relate the pose increments $\delta\omega$ (rotation) and ΔT (translation) to the computed measurement errors using the current pose estimate. At each iteration in the minimization process, the linear system of equations is solved to find the best increment vector. This increment is added to the current pose estimate and the process repeated until there is convergence.

In the above system of equations, (η_x, η_y) represents the measurement noise. If the correct estimate of pose were known, η_x and η_y would be equal to the sum of the measurement error of the image point location and the projection of the error in the model point along the image x-axis and y-axis respectively. The measurement of the image point location is assumed to be corrupted with zero-mean independent gaussian noise. In our case, for lack of any other knowledge, it is assumed that the noise in the measurements is independent across all points and is also the same. The 3D model points are also assumed to be corrupted by zero-mean independent gaussian noise. Therefore in the "2m" system of linear equations, the noise in the two equations for every point is correlated. Thus the covariance matrix "V" corresponding to the noise in the linear system of equations (18) in the Appendix is a band matrix in which the non-zero entries are (2 x 2) matrices about the diagonal. The output covariance matrix for the pose rotation and translation parameters is given by equation (20) evaluated at the final pose estimate.

Using the formula for the best linear unbiased estimate described in equation(19) in the Appendix, the formula for the pose increment at any iteration is derived. If the model noise was zero and the noise in the image measurements were assumed to be same for all points, then the input covariance matrix would be an identity matrix scaled by the standard deviation of image noise.

3 Induced Stereo

In this section, we present techniques for computing 3D estimates of new points in the world coordinate system from their tracked image locations over a multi-frame sequence. The mathematics for both extending the model and refining the initial modeled points is presented. Computed with the estimate of each new model point is an estimate of the covariance of its error. These covariances are functions of the input image measurement covariances and the initial 3D model point covariances.

Image features (both new features and modeled image features appearing in the images) are tracked over a sequence of frames using the computed optic flow between pairs of successive frames [15]. Typically corners (defined by the intersection of two image lines) are tracked although any image feature which can be reliably tracked may be used. The initial matching of image features to the partial model for the first frame may be done by a matching process such as in [4]. Combining the results of the initial matching and the feature tracking, correspon-

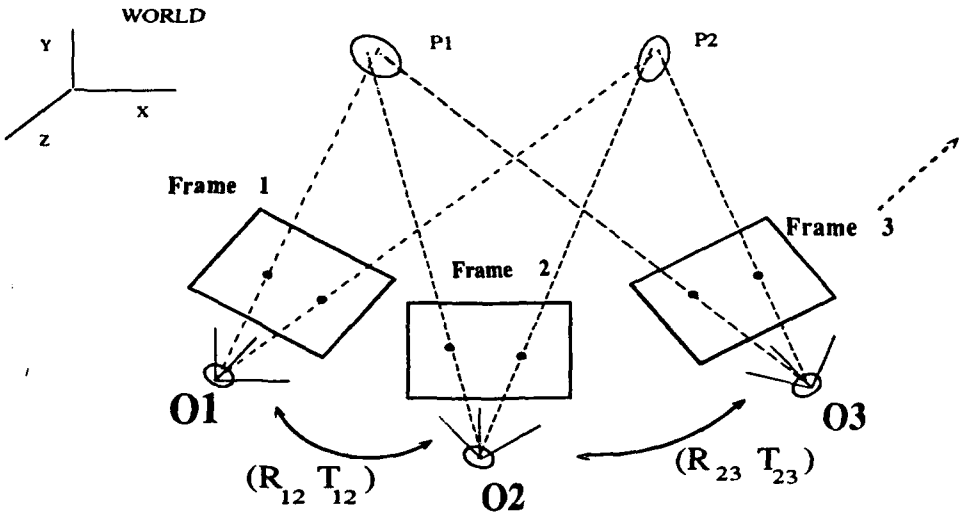


Figure 1: Model Extension and Refinement.

dences between image features and the partial model for each frame are established. Using these correspondences, pose estimation is done for each frame using the method presented in the previous section.

The image projection ray for an image point in a particular frame is defined as the ray originating from that frame's optic center and passing through the image point. Given the pose estimates for each frame, the vectors corresponding to these projection rays in the world coordinate system can be obtained. The 3D estimate of the point is the pseudo-intersection of all the image projection rays for a tracked image point. In order to combine 3D measurements from a sequence of frames, a stable coordinate frame should be used; a nice property of the system described here is that the pose estimation process provides the world coordinate frame as this stable coordinate frame. Independent measurements can be made relating the coordinate system of each frame in the sequence to the world coordinate frame.

Points are located by the psuedo-intersection process in two steps. In the first step, a 3D error function is minimised to find an initial estimate of the point's location. This step, however, does not yield the optimal estimate since the various error terms are not weighted by the input covariances. In the second step, an image-based error function is optimised in which the error terms are inversely weighted by a combination of the input covariances of the pose estimate and the image measurements.

Let r_i be the unit vector corresponding to the image projection ray for an image point in the i'th frame. The pose estimation for this frame is given by the rotation R_i and translation $\vec{T}_i$ (see equation (1)). Since the image projection rays do not intersect at a unique point[2], the 3D pseudo-intersection point p_i is obtained by minimising an error function E:

$$E = \sum_{i=1}^{n} \|(R_i(\vec{p}) + \vec{T}_i) \times r_i\|^2 \qquad (11)$$

Therefore the 3D error function E (used in the first step) is the sum of squares of the perpendicular distances from the psuedo-intersection point $\vec{p}$ to the image projection rays. Differentiating E with respect to the unknown variable $\vec{p}$ leads to a set of linear equations, which are then solved to give the initial estimate for $\vec{p}$.

In the second step, the pose constraint equations (2, 3) are used to formulate image-based error equations for the X and Y projections of the model points.

$$\frac{1}{p_{cz}}\vec{C}_{xi} \cdot R_i(\vec{p}) = -\frac{1}{p_{cz}}\vec{C}_{xi} \cdot \vec{T}_i + \zeta_X \qquad (12)$$

$$\frac{1}{p_{cz}}\vec{C}_{yi} \cdot R_i(\vec{p}) = -\frac{1}{p_{cz}}\vec{C}_{yi} \cdot \vec{T}_i + \zeta_Y \qquad (13)$$

where ζ_X and ζ_Y are the noise terms in the two equations. ζ_X and ζ_Y are functions of both noise in pose $\Delta\vec{T}_i$ and $\delta\vec{\omega}_i$ and image noise $(\Delta X, \Delta Y)$:

$$\zeta_X = \Delta X + \frac{1}{p_{cz}}\vec{C}_{xi} \cdot \Delta\vec{T}_i + \frac{1}{p_{cz}}\delta\vec{\omega}_i \cdot \vec{b}_i \qquad (14)$$

$$\zeta_Y = \Delta Y + \frac{1}{p_{cz}}\vec{C}_{yi} \cdot \Delta\vec{T}_i + \frac{1}{p_{cz}}\delta\vec{\omega}_i \cdot \vec{b}_i \qquad (15)$$

In this case the 3D model point $\vec{p}$ is the unknown variable. The denominator p_{cz} in the equations (12 and 13) corresponds to the depth of the point and is a function of the unknown variable $\vec{p}$. Therefore for each frame over which the point is tracked, two non-linear constraint equations (12 and 13) are obtained [3]. An iterative procedure is employed to solve the system of non-linear equations. At each iteration, the denominator p_{cz} is held

[2] Due to noise both in image measurements and pose estimates.

[3] A minimum of two frames is needed to solve the system of equations.

constant using the previous estimate of $\vec{p}$ and the resulting linear system of equations is solved using equation (19) (see Appendix). The iterative procedure is repeated until there is convergence. In practice, we have found one iteration is sufficient for robust results. The input covariance matrix V required for in equation (19) is obtained from the expressions derived above for the noise terms ζ_X, ζ_Y. The output covariance of the 3D point estimate is given by equation (20) in the Appendix.

In the batch method, information from all frames is used simultaneously to estimate the 3D locations of tracked image points. However, it may be desired to sequentially update the location of new points after every pair (or a larger set) of frames. In the sequential or quasi-batch mode, equations (12 and 13) are again used to estimate the 3D location of image points tracked over the current set of frames. However, these new estimates must be fused with the previous estimates to obtain the current optimal estimate. Associated with each estimate is a covariance matrix representing the uncertainty in the estimate. These covariance matrices are used to fuse the two estimates and provide a new uncertainty matrix using the standard Kalman Filtering equations.

Let the estimate of the point's 3D location and its covariance at frame "t_1" be $\vec{p}(t_1)$ and $\Lambda_p(t_1)$ respectively. A new 3D location measurement $\vec{Q}$ with uncertainty (covariance matrix Λ_Q) is computed from a batch of "n" image frames. The fused location estimate $\vec{p}(t_n)$ and updated covariance matrix $\Lambda_p(t_n)$ at frame "t_n" are given by:

$$\vec{p}(t_n) = \Lambda_p(t_n)(\Lambda_p(t_1)^{-1}\vec{p}(t_1) + \Lambda_Q^{-1}\vec{Q}) \quad (16)$$

$$\Lambda_p(t_n) = (\Lambda_p(t_1)^{-1} + \Lambda_Q^{-1})^{-1} \quad (17)$$

This same method is used for model refinement. Initial model points have associated with them their input covariance matrices. When the model is tracked over a new batch of frames, 3D measurements can also be made for the model points by the above psuedo-intersection procedure. These new measurements are fused with the old estimate using the above equation.

3.1 Model Extension and Refinement Algorithm

The algorithm for model extension and refinement using a current batch size of "n" ($n \geq 2$) frames can be summarized as follows:

Step 1 Given a partial 3D model and an image, establish correspondences between model points and image points using a matching technique such as in [4].

Step 2 Track image points over the batch of "n" frames using the computed optic flow between successive pairs of images [15].

Step 3 Using the correspondences established above between model points and image points, compute the pose for each image frame using the method described in Section 4.2.

Step 4 Estimate the 3D location of both new points and initial model points in world coordinates using the two-step approach developed in Section 4.3 and the feature correspondences established in Step 2 for the current batch of "n" frames.

Step 5 Fuse initial estimates of both the new points and the model points with any previous estimates using equations (16,17).

4 Experimental Results

This section presents experimental results of applying the model extension and refinement algorithms to two real data image sequences. Figures 2 and 4 show example images from the BOX and A211 sequences respectively. In all experiments the image center was assumed to be at the center of the image frame and the effective focal length was calculated from the manufacturers specification sheets. In another paper [11], we have shown that errors in the image center do not significantly affect the location of new points in a world coordinate system.

4.1 Box Sequence

The BOX sequence was generated by rotating the box (in Fig. 2) about its central vertical axis while the camera was kept stationary. Consecutive images in the sequence were taken after a rotation of approximately 3.6 degrees. In the first frame, the camera was about 650 mm distant from the top front corner of the box. The location of 30 points (marked in Fig.2 by circles and crosses) in a world coordinate system was measured to an accuracy of approximately 1 mm along each axis. The depth of the points (in the first frame's coordinate system) used in our experiment varied from 575 mm to 700 mm. The thirty points were tracked over the set of 8 frames.

The fifteen points marked by crosses in Figure 2 were used as the initial model for pose estimation [10] in each frame. Various experiments were performed with different amounts of synthetic uniform noise added to the measured 3D locations of the cross points. Using the computed poses, 3D estimates of the remaining 15 points (marked by circles in Figure 2) were computed. In addition, the initial model of 15 (cross marked) points was refined. The algorithm described in Section 4.3.1 was run in a batch mode over all 8 frames to perform these experiments; the results are reported in Table 1. The first column of Table 1 gives the range of noise added to the initial model points. Thus a 10 mm entry in the first column means uniform noise in the range of +/- 10 mm was added to each of the 3D coordinates of the model points. The average error[4] of the 15 initial model points for each experiment (prior to any refinement) is given in the second column of Table 1. The third column in the table shows the results of the model refinement process; it gives the average output error of the 15 (now refined) initial model points. The fourth column in the table shows the results of the model extension process;

[4]The average error is the root mean square (RMS) value of the 3D location error of all points.

it gives the average output error of the 15 new (circle) points.

As can be seen from the first row in Table 1, the average error for model extension when there is no noise in the initial model is 1.38 mm. The maximum error was 2.6 mm and the minimum error was 0.44 mm. The average percentage error was 0.25 %. The percentage error is calculated by dividing the absolute 3D error by the depth of the point from the origin of the camera in the first image's coordinate frame. As the noise in the initial model increases, the errors in model extension and refinement also increases. However, except for the first two cases in Table 1, the average output error for both model extension and refinement were significantly lower than the average input error of the initial model points.

The model extension and refinement algorithm was also run in a sequential mode, where new 3D locations were computed after every new pair of frames and the results were fused with previous estimates. Figure 3 shows the results of such an experiment. For this experiment, the range of input noise was 5mm and the average error of the initial model points was 4.49 mm (corresponding to the fifth row in Table 1). The average output error in location of both the initial model points and the new (circle) 3D points is plotted for every image frame in the sequence. As can be seen in the figure, the 3D error in both the initial model points and the unknown points monotonically decreases across all frames. The average error of the new points is reduced from 6.5 mm after the first pair of frames to about 3.7 mm at the end. The average error of the initial points is reduced from 4.49 mm to about 2.8 mm.

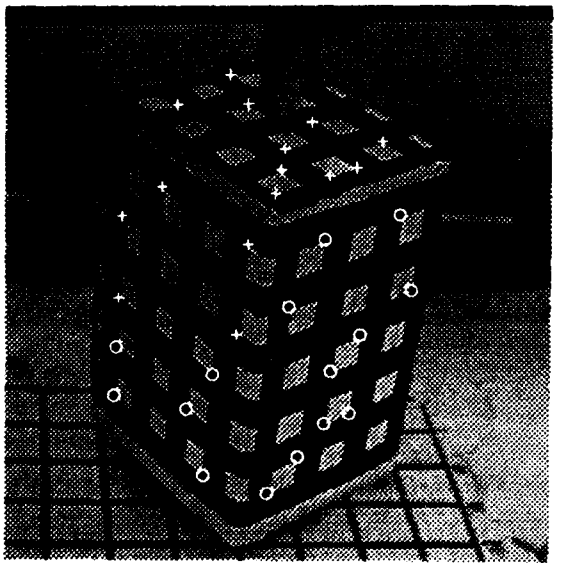


Figure 2: Box Image. The points marked by crosses were used to compute the 3D pose for each frame. Using these poses, the 3D location of the numbered points marked by circles is computed.

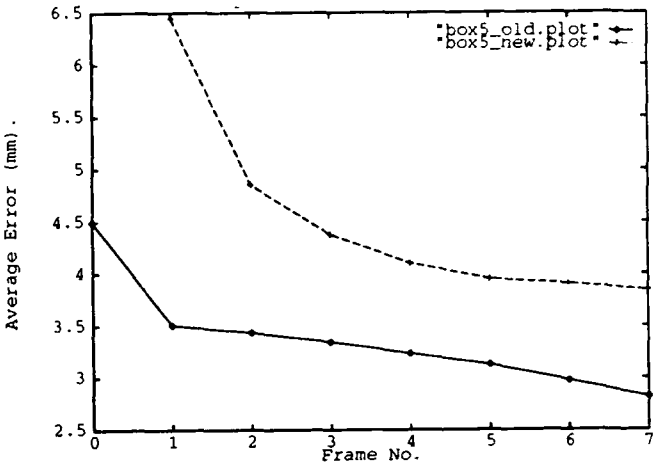


Figure 3: Box Sequence. Plot of average error over the frame sequence for for the new points (Model Extension) and for the initial model points (Model Refinement).

4.2 A211 sequence

The A211 sequence was generated by taking images from a camera mounted on a mobile robot. The robot was translated roughly along the optical axis of the camera and 10 image frames were taken after every 0.38 feet. Thus the total translation of the camera was 3.42 feet. Figure 4 shows the first frame in the image sequence. Objects in the scene ranged from 8 feet to 20 feet away in the first image frame.

The initial model in this experiment was built using Sawhney's [14] algorithm for segmenting and locating shallow structures[5]. Seven points (the points marked by

[5]Shallow structures are those whose extent in depth is small compared to their average depth from the camera.

Table 1: **Computed average output 3D location errors for model extension process with noisy input model points for the Box Sequence of 8 frames.** Input Noise to model is synthetic uniform noise.

Range Input Noise	Average Input Noise	Average Output Noise	
		Initial Points	New Points
mm	mm	mm	mm
0	0.00	0.00	1.38
1	1.02	1.01	1.69
2	1.95	1.52	1.92
3	3.06	2.00	2.23
5	4.49	3.00	3.78
7	6.96	3.32	3.84
10	10.25	4.16	6.31
20	17.29	10.32	16.23

crosses in Figure 4) lying on shallow structures recovered by this algorithm were used as the initial model points. The 3D model locations were constructed by extending the image projection rays in the first image's coordinate frame of the seven points to the depth computed by Sawhney's algorithm. Thus, the model coordinate frame is the same as the first image's coordinate frame.

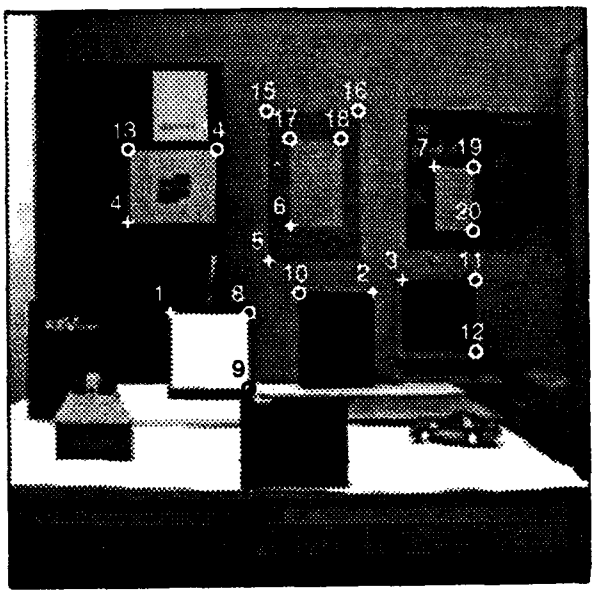


Figure 4: A211 Image. The points marked by crosses were used to compute the 3D pose for each frame. Using these poses, the 3D location of the numbered points marked by circles is computed.

The model extension and refinement algorithm was run in a sequential mode. Table 2 shows the result of locating the 13 new points (circled and numbered from 8 to 20 in the Figure 4) and refining the seven initial model points. The ground truth available for the experiment was only the depths (as opposed to 3D location) of the points in the first image's coordinate frame. Thus the results shown in Table 2 compare the measured depth value (ground truth) with the recovered depth value. Column 2 in the table shows the measured depth of the point in the first image coordinate frame. Columns 3 and 4 show the input error and percentage error in depth (before model refinement and extension) respectively. Thus, for the new points (Nos. 8 to 20) these two columns are blank, since no prior estimate is assumed for them. Columns 5 and 6 show the input error and percentage error in depth (after model refinement and extension) respectively. The percentage error in depth is computed with respect to the depth in the first image's coordinate frame.

The average input error in depths of the seven model points was 0.4 feet (1.85 % error). At the end of the ten frames, the average error of the 7 initial points was 0.37 feet (1.76 %). The thirteen new points were located to an average accuracy of 0.4 feet (1.63 %). Thus, in this

Table 2: Absolute and Percentage 3D location errors for points in A211 sequence (see Fig. 4.)

		INPUT		OUTPUT	
Pt. No.	Depth ft.	Abs. Err. ft.	% Err.	Abs. Err. ft.	% Err.
Initial Points					
1	13.4	0.24	1.80 %	0.24	1.78 %
2	14.6	0.19	1.31 %	0.20	1.34 %
3	19.0	0.74	3.88 %	0.66	3.46 %
4	19.0	0.16	0.86 %	0.11	0.60 %
5	20.4	0.13	0.62 %	0.17	0.86 %
6	20.4	0.39	1.90 %	0.32	1.60 %
7	20.4	0.49	2.38 %	0.46	2.25 %
New Points					
8	13.4	-	-	0.11	0.79 %
9	13.4	-	-	0.00	0.01 %
10	14.6	-	-	0.53	3.65 %
11	19.0	-	-	0.73	3.86 %
12	19.0	-	-	0.54	2.82 %
13	19.0	-	-	0.11	0.59 %
14	19.0	-	-	0.07	0.34 %
15	20.4	-	-	0.23	1.13 %
16	20.4	-	-	0.27	1.32 %
17	20.4	-	-	0.12	0.57 %
18	20.4	-	-	0.34	1.65 %
19	20.4	-	-	0.62	3.02 %
20	20.4	-	-	0.59	2.92 %

experiment there was only slight improvement for the model refinement process; however the model extension process was fairly accurate in locating new points. If the initial model given to the model extension process is noise free, then the average error in recovering the thirteen new points is 0.2 feet (0.94 %).

The robust recovery of the location of new 3D points depends on the camera motion. Optimal angles for triangulation are achieved when there is significant translation parallel to the image plane. In the A211 sequence, the translation of the camera is mostly along the optical axis. Thus, the FOE (focus of expansion) lies on the image plane. Points close to the FOE have hardly any disparity and their depths cannot be reliably estimated. In the BOX experiment, the higher accuracy with which 3D parameters of the new points were computed is due primarily to the fact that a large component of the motion over the sequence is approximately parallel to the image plane. Such motion is best for accurate triangulation.

5 Conclusions

The techniques presented in this section are preliminary efforts for model extension and refinement of point data. The experimental results show that knowledge of a few points can greatly increase the accuracy of 3D recovery in comparision to traditional algorithms from motion and stereo analysis. However, the accuracy of the model ex-

tension process depends on the initial accuracy of the model points. To make the system less sensitive to the initial accuracy of the model points, one possible solution would be to couple methods of motion analysis with those of pose recovery.

If the initial model points have a large amount of noise, then the poses determined for any batch of frames will be highly correlated. In this case, the 3D location estimates of new points will be correlated both across all points and also all frames. To fully account for this correlation, covariance matrices equal to the size of number of points times number of frames will have to be inverted. In our case, it is assumed that the initial points do not have significant noise and hence the cross-correlations can be ignored. But for larger amounts of noise, it may not be possible to ignore these effects. These cross-terms are exactly what Oliensis and Thomas [13] incorporate in their motion analysis paper.

Finally, the terms model extension and refinement are slightly abused in this paper. Model extension and refinement are not limited to just locating new points in the scene. Ultimately, it is desired to build 3D surface and volumetric models and integrate the new 3D measurements with the existing higher order models; this has been left for future work.

Appendix

Some facts from linear system estimation theory are reviewed. An unknown parameter vector $\vec{x}$ with "p" elements is related to a set of "n" noisy observations $\vec{y}$ by the following equation:

$$A\vec{x} = \vec{y} + \vec{\eta} \tag{18}$$

where $\vec{\eta}$ is zero-mean Gaussian noise with covariance matrix V. Assume, that this set of equations is an overconstrained system. Then the Best Linear Unbiased Estimate (BLUE) of the unknown vector $\vec{x}$ is given by:

$$\hat{x} = (A^T V^{-1} A)^{-1} A^T V^{-1} y \tag{19}$$

The covariance matrix "P" of the output parameters is given by:

$$P = (A^T V^{-1} A)^{-1} \tag{20}$$

References

[1] G. Adiv, *Interpreting Optical Flow*, PhD thesis, COINS Tech. Report 85-35, Univ. Of Mass. at Amherst, MA., 1985.

[2] P. Anandan, *Measuring Visual Motion from Image Sequences*, PhD Thesis, COINS Tech. Report TR 87-21, Univ. Of Mass. at Amherst, MA., 1987.

[3] N. Ayache and O.D. Faugeras, "Building, Registrating and Fusing Noisy Visual Maps," *The International Journal of Robotics Research*, Vol. 7, No. 6, Dec. 1988.

[4] J. R. Beveridge, R. Weiss and E. Riseman, "Optimisation of 2-Dimensional Model Matching," *IEEE International Conference on Pattern Recognition*, Atlantic City, N.J., June 1990.

[5] T. J. Broida and R. Chellappa, "Estimating the Kinematics and Structure of a Rigid Object from a Sequence of Monocular Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 497–513, 1991.

[6] S. Chandraskekhar and R. Chellapa, "A Two-Step Approach to Passive Navigation Using a Monocular Image Sequence," *USC-SIPI Technical Report 170*, University of Southern California, Electrical Engineering-Systems, 1991.

[7] N. Cui, J. Weng and P. Cohen, "Extended structure and motion analysis from monocular image sequences," *Proceedings 3rd IEEE International Conference on Computer Vision*, Osaka, Japan, Dec. 1990.

[8] R. Dutta and M. Snyder, "Robustnesss of Correspondence-Based Structure from Motion," *Proceedings 3rd IEEE International Conference on Computer Vision*, Osaka, Japan, Dec. 1990.

[9] B. K. P. Horn, "Relative Orientation," *International Journal of Computer Vision*, Vol. 4, pp. 59-78, 1990.

[10] R. Kumar and A.R. Hanson, "Robust Estimation of Camera Location and Orientation from Noisy Data with Outliers," *Proc. IEEE Workshop on Interpretation of 3D scenes*, Austin, Texas, Nov. 1989.

[11] R. Kumar and A.R. Hanson, "Sensitivity of pose refinement to accurate estimation of camera parameters," *IEEE International Conference on Computer Vision*, Osaka, Japan, Dec. 1990.

[12] L. H. Matthies, *Dynamic Stereo Vision*, Ph.D. thesis, Carnegie Mellon University, Oct. 1989.

[13] J. Oliensis and J. I. Thomas, "Incorporating motion error in multi-frame structure from motion", *Proceedings IEEE Workshop on Visual Motion*, Princeton, N.J., Oct. 1991.

[14] H. S. Sawhney and A. R. Hanson, "Identification and 3D description of 'shallow' environmental structure in a sequence of images", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 179–186, Hawaii, June 1991.

[15] L. R. Williams and A. R. Hanson, "Translating Optical Flow into Token Matches and Depth from Looming", *Second Int. Conf. on Computer Vision*, pp. 441-448, 1989.

[16] Z. Zhang and O. D. Faugeras, "Building a 3D World Model with a Mobile Robot: 3D Line Segment Representation and Integration," *IEEE International Conference on Pattern Recognition*, Atlantic City, N.J., June 1990.

Posterior Marginal Pose Estimation

William M. Wells III *
Massachusetts Institute of Technology
Artificial Intelligence Laboratory
545 Technology Square, Cambridge, Massachusetts 02139

Abstract

A method of evaluating hypotheses about the pose of a known object in an image, posterior marginal pose estimation, is described. A detailed metrical object model is assumed. A probabilistic model of image features is combined with a simple prior on both the pose and the feature interpretations to yield a pose objective function. The parameters that appear in the probabilistic models can be derived from images in the application domain. By extremising the objective function, an estimate of the pose of the object in the image results.

Within this framework, good models of feature uncertainty allow for robustness despite inaccuracy in feature detection. In addition, the relative likelihood of features arising from either the object or the background can be evaluated in a rational way. It captures important aspects of recognition: the amount of the image that is explained in terms of the model, as well as the metrical consistency of the hypothesis. It also allows these two aspects to be traded off in a rational way based on domain statistics.

The objective function takes a simple form when feature deviations are modeled by normal densities and the projection model is linear. Several linear projection and featu e models are discussed.

A preliminary evaluation of the posterior marginal pose estimation objective function in a domain of synthetic range discontinuity features is described. In that domain the objective function has a prominent sharp peak near the correct pose. Some local maxima are also apparent. Relation to other work, and possible extensions are discussed.

*This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by an Office of Naval Research University Research Initiative grant under contract N00014-86-K-0685, and in part by the Advanced Projects Agency of the Department of Defense under Army contract number DACA76-85-C-0010 and under Office of Naval Research contract N00014-85-K-0124. Summer support was provided by Group 53 at the Massachusetts Institute of Technology, Lincoln Laboratory.

1 Introduction

Vision problems are sometimes usefully posed as optimization problems where solutions to an objective function are sought. Such an approach is used by Barnard in stereo matching [1], Blake and Zisserman [2] in image restoration and Beveridge, Weiss and Riseman [3] in line segment based model matching. One advantage of an objective function approach is that algorithm design can be separated from the specification of the computation to be performed. Useful objective functions may be designed based on ad-hoc considerations. In such cases, plausible forms for components of the objective function are often summed using trade-off parameters. These trade-off parameters are determined either empirically or by guess.

Statistical estimation has proven useful as a theoretical framework for deriving objective functions for some areas of vision. The work of Yuille, Geiger and Bülthoff on stereo [4] is one example, while the work of Geman and Geman on image restoration [5] is another. One advantage of deriving objective functions from statistical theories is that the forms of the objective function components are clearly related to specific probabilistic models. Another advantage is that the trade-off parameters can then be derived from measurable statistics of the domain.

In a previous paper on MAP model matching [6] the model matching problem was posed as an optimization problem resulting from a statistical theory. There the solution to the matching problem was the set of parameters of the occurrence of some known object in an image. These parameters were the position and orientation, or pose, of the object, as well as the correspondences between model and image features. The method was shown to provide effective evaluations of match and pose hypotheses. Tree search was explored as a mechanism for the optimization problem. This route has proved to be computationally burdensome.

The research reported here is aimed at easing the computational cost of the search. The idea is to generate candidate starting configurations for the search, as a way of limiting the total amount of the search space that is explored. The method involves looking first for candidate

poses to seed the tree search over match and pose. This sort of approach has been used in previous recognition systems, see the work of Grimson [7] for an example.

In the next section the basics of the MAP model matching theory are reviewed, the following section then extends the development to posterior marginal pose estimation.

2 MAP Model Matching

In this section an objective function for model matching will be derived using a MAP criterion.

Briefly, probability densities of image features, conditioned on the parameters of match and pose ("the parameters"), are combined with prior densities on the parameters using Bayes' rule. The result is a posterior probability density on the parameters, given an observed image. An estimate of the parameters is then formulated by choosing them so as to maximise their a-posteriori probability. (Hence the term *MAP*. See Beck and Arnold's textbook [8] for a discussion of MAP estimation.) MAP estimators are especially practical when used with normal probability densities.

In this section the domain is matching among generalized v-dimensional point features. (Section 4 discusses several such models.) Specific probability densities are assumed here in the interest of simplicity and concreteness; image features are assumed to be mutually independent; and matched image features are assumed to be normally distributed about their predicted positions in the image.

Additionally, unmatched (background) features are assumed to be uniformly distributed in the image. These densities are combined with a simple prior on the parameters. A linear projection model is introduced and a simple MAP estimator results.

Let the image that is to be analysed be represented by a set of v-dimensional point features

$$Y = \{Y_1 Y_2 \dots Y_n\} \quad , \quad Y_i \in R^v \quad .$$

The model to be matched is also described by a set of point features, these are represented by real matrices:

$$M = \{M_1 M_2 \dots M_m\} \quad .$$

Section 4 describes examples of this type of representation.

The parameters to be estimated in matching are the correspondences between image and model features, and the pose of the model instance in the image. The correspondences are described by an interpretation vector

$$\Gamma = [\Gamma_1 \Gamma_2 \dots \Gamma_n] \quad , \quad \Gamma_i \in M \cup \{\perp\} \quad .$$

Here $\Gamma_i = M_j$ means that image feature i corresponds to model feature j, and $\Gamma_i = \perp$ means that image feature i is due to the background.

The pose of the model instance in the image, β, is a real vector. An associated projection function P is defined:

$$P(M_i, \beta) \in R^v \quad .$$

P maps model features into the v-dimensional image according to the model's pose.

Based on the above, the probability density function on image features may be written

$$p(Y_i \mid \Gamma, \beta) = \begin{cases} \frac{1}{W_1 W_2 \dots W_v} & \text{if } \Gamma_i = \perp \\ N_{\psi_i}(Y_i - P(\Gamma_i, \beta)) & \text{otherwise} \end{cases} \quad (1)$$

where $N_{\psi_i}(x) = (2\pi)^{-\frac{v}{2}} |\psi_i|^{-\frac{1}{2}} \exp(-\frac{1}{2} x^T \psi_i^{-1} x)$. Here ψ_i is the covariance matrix associated with image feature i. Thus image features arising from the background are uniformly distributed over the space of the image (the width of the image space along dimension i is given by W_i), and matched image features are normally distributed about their predicted locations in the image. In some applications ψ could be a constant – an assumption that the feature statistics are stationary in the image.

Assuming independent features we have

$$p(Y \mid \Gamma, \beta) = \prod_i p(Y_i \mid \Gamma, \beta)$$

$$= \prod_{\Gamma_i = \perp} \frac{1}{W_1 W_2 \dots W_v} \prod_{\Gamma_i \neq \perp} N_{\psi_i}(Y_i - P(\Gamma_i, \beta)) \quad .$$

Next, a simple prior probability density function is defined on the pose and interpretation. The probability that a image feature belongs to the background is B; the remaining probability is uniformly distributed for correspondences to the m model features.

$$p(\Gamma_i) = \begin{cases} B & \text{if } \Gamma_i = \perp \\ \frac{1-B}{m} & \text{otherwise} \end{cases} \quad . \quad (2)$$

Prior information on the pose is assumed to be supplied as a normal density,

$$p(\beta) = N_{\psi_\beta}(\beta - \beta_0)$$

where $N_{\psi_\beta}(x) = (2\pi)^{\frac{-z}{2}} |\psi_\beta|^{-\frac{1}{2}} \exp(-\frac{1}{2} x^T \psi_\beta^{-1} x)$. Here ψ_β is the covariance of the pose estimate and the dimensionality of β is denoted by z. With this choice for the form of the pose prior the system is closed in the sense that the resulting pose estimate will also be normal. This is convenient for coarse-fine, as discussed in Section 8. If little is known about the pose a-priori, the prior may be made quite broad. This is expected to often be the case.

Then assuming independence of the correspondences and the pose (before the image is seen), the composite prior is

$$p(\Gamma, \beta) = N_{\psi_\beta}(\beta - \beta_0) \prod_{\Gamma_i = \perp} B \prod_{\Gamma_i \neq \perp} \frac{1 - B}{m} \quad .$$

Now using Baye's rule, the a-posteriori probability density of Γ and β may be calculated:

$$p(\Gamma, \beta \mid Y) = \frac{p(Y \mid \Gamma, \beta) p(\Gamma, \beta)}{C} \quad ,$$

where C is a normalising constant independent of Γ and β.

The MAP strategy is used to obtain estimates of the correspondences and pose by maximising the posterior density with respect to Γ and β, as follows

$$\widehat{\Gamma, \beta} = \arg\max_{\Gamma, \beta} p(\Gamma, \beta \mid Y) \quad .$$

3 Posterior Marginal Pose Estimation

The following method was motivated by the observation that, in tree searches of the objective function of MAP model matching, hypotheses having "poor" matches scored poorly in the objective function. The implication was that summing posterior probability over all matches (at a specific pose) might provide a good pose evaluator. This has proven to be the case in the experiment described in Section 6. Additional motivation was provided by the work by Yuille, Geiger and Bülthoff on stereo [4]. They discussed computing disparities in a statistical theory of stereo where a marginal is computed over matches.

The essence of posterior marginal pose estimation is to choose the pose that maximizes the posterior probability density of the pose, given an image:

$$\hat{\beta} = \arg\max_{\beta} p(\beta \mid Y) \quad .$$

The posterior probability density of the pose is computed from the joint posterior probability on pose and match, by taking the marginal over the possible matches:

$$p(\beta \mid Y) = \sum_{\Gamma} p(\Gamma, \beta \mid Y) \quad .$$

Using Bayes' rule, as in Section 2, the posterior marginal may be written as

$$p(\beta \mid Y) = \sum_{\Gamma} \frac{p(Y \mid \Gamma, \beta) p(\Gamma, \beta)}{C} \quad .$$

Next, the models for the joint probability densities of image features, Y, and the priors for Γ and β, as described in Section 2, are used to express the posterior marginal of β in terms of the component densities:

$$p(\beta \mid Y) = \frac{1}{C} \sum_{\Gamma_1} \cdots \sum_{\Gamma_n} \prod_i p(Y_i \mid \Gamma, \beta) \prod_i p(\Gamma_i) p(\beta)$$

or

$$p(\beta \mid Y) = \frac{p(\beta)}{C} \sum_{\Gamma_1} \cdots \sum_{\Gamma_n} \prod_i [p(Y_i \mid \Gamma_i, \beta) p(\Gamma_i)] \quad .$$

Breaking one factor out of the product gives

$$p(\beta \mid Y) = \frac{p(\beta)}{C} \sum_{\Gamma_1} \cdots \sum_{\Gamma_n} \prod_{i=1}^{n-1} [p(Y_i \mid \Gamma_i, \beta) p(\Gamma_i)] \, p(Y_n \mid \Gamma_n, \beta) p(\Gamma_n) \, ,$$

or

$$p(\beta \mid Y) = \frac{p(\beta)}{C} \sum_{\Gamma_1} \cdots \sum_{\Gamma_{n-1}} \prod_{i=1}^{n-1} [p(Y_i \mid \Gamma_i, \beta) p(\Gamma_i)] \sum_{\Gamma_n} p(Y_n \mid \Gamma_n, \beta) p(\Gamma_n) \, .$$

Continuing in similar fashion yields

$$p(\beta \mid Y) = \frac{p(\beta)}{C} \prod_i \left[\sum_{\Gamma_i} p(Y_i \mid \Gamma_i, \beta) p(\Gamma_i) \right] \quad .$$

Splitting the Γ_i sum into its cases,

$$p(\beta \mid Y) = \frac{p(\beta)}{C} \prod_i \Big[p(Y_i \mid \Gamma_i = \perp, \beta) p(\Gamma_i = \perp) + \sum_{M_j} p(Y_i \mid \Gamma_i = M_j, \beta) p(\Gamma_i = M_j) \Big] \quad .$$

Now substituting the densities assumed in the model of Section 2 in Equations 1 and 2 yields

$$p(\beta \mid Y) = \frac{p(\beta)}{C} \prod_i \Big[\frac{1}{W_1 \cdots W_v} B + \sum_{M_j} N_{\psi_i}(Y_i - P(M_j, \beta)) \frac{1 - B}{m} \Big] \quad .$$

Re-arrangement gives

$$p(\beta \mid Y) = \left(\frac{B}{W_1 \cdots W_v} \right)^n \frac{p(\beta)}{C} \prod_i \left[1 + \sum_{M_j} \frac{W_1 \cdots W_v}{m} \frac{1 - B}{B} N_{\psi_i}(Y_i - P(M_j, \beta)) \right]$$

Next the objective function for posterior marginal pose estimation is defined as the scaled logarithm of the posterior marginal probability of the pose, as follows,

$$L(\beta) = \ln \left[\left(\frac{W_1 \cdots W_v}{B} \right)^n C \, p(\beta \mid Y) \right] \quad ,$$

or, finally,

$$L(\beta) = \ln p(\beta) + \sum_i \ln \left[1 + \sum_{M_j} \frac{W_1 \cdots W_v}{m} \frac{1 - B}{B} N_{\psi_i}(Y_i - P(M_j, \beta)) \right] . \quad (3)$$

4 Linear Projection and Feature Models

As noted above, the objective function assumes a simple form when the projection model is linear in the parameters of the projection. Two different feature and projection models that meet this criterion are described here. They share a model of projection, rotation, and scaling in the 2-D plane. The features are local and could represent fragments of extended features such as curves. Methods for 3-D are discussed.

4.1 2-D Point Feature Model

A linear projection model takes the following form:

$$\eta_i = P(M_i, \beta) = M_i \beta \quad . \tag{4}$$

The pose of the object is represented by β, the model feature i by M_i, and η_i is the projection of the model feature into the image by pose β.

One 2-D point feature and projection model is defined by

$$\eta_i = \begin{bmatrix} p'_{ix} \\ p'_{iy} \end{bmatrix} \qquad M_i = \begin{bmatrix} p_{ix} & -p_{iy} & 1 & 0 \\ p_{iy} & p_{ix} & 0 & 1 \end{bmatrix}$$

$$\text{and} \qquad \beta = \begin{bmatrix} \mu \\ \nu \\ t_x \\ t_y \end{bmatrix} \quad .$$

The coordinates of model point i are p_{ix} and p_{iy}. The coordinates of the model point i, projected into the image by pose β, are p'_{ix} and p'_{iy}. This transformation is equivalent to a translation by T, rotation by θ, and scaling by s where

$$T = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \qquad s = \sqrt{\mu^2 + \nu^2} \qquad \theta = \arctan(\frac{\nu}{\mu}) \quad .$$

This model was used by Faugeras and Ayache in their vision system HYPER [9]. This representation has an un-symmetrical way of representing the two classes of features, which seems odd due to their essential equivalence, however the trick facilitates the linear formulation of projection given in Equation 4.

4.2 2-D Oriented-Range Feature Model

A 2-D projection and feature model that incorporates local information about the coordinates, normal, and range at a point along a curve of range discontinuity, is defined by

$$\eta_i = \begin{bmatrix} p'_{ix} \\ p'_{iy} \\ c'_{ix} \\ c'_{iy} \end{bmatrix} \qquad M_i = \begin{bmatrix} p_{ix} & -p_{iy} & 1 & 0 \\ p_{iy} & p_{ix} & 0 & 1 \\ c_{ix} & -c_{iy} & 0 & 0 \\ c_{iy} & c_{ix} & 0 & 0 \end{bmatrix} \quad .$$

The point coordinates and β are as above. c_{ix} and c_{iy} are a vector who's direction is perpendicular to the range discontinuity and pointing away from the discontinuity, while the length of the vector is the inverse of the range at the discontinuity. The counterparts in the image are given by c'_{ix} and c'_{iy}. The aggregate feature translates, rotates and scales correctly.

4.3 Linear 3-D Projection Models

The technique used in the above two methods amounts to making linear combinations of the model with a copy rotated 90 degrees in the plane.

In their paper, "Recognition by Linear Combination of Models" [10], Ullman and Basri describe a scheme for synthesising views under 3D orthography with rotation and scale that has a linear parameterization. Their method may prove useful within the framework described here.

5 Feature Deviation Models

In this section, feature model stationarity and normal feature deviation models are discussed.

In the formulation presented in Sections 2 and 3, the image feature covariance depends only on the image feature index. A simple extension would also index the covariance on the corresponding model feature in the interpretation. This becomes a useful method when the image counterparts of some model features are known to fluctuate more than others. In the experiments a form of stationarity was used. This is described next.

5.1 Oriented Stationary Statistics

The image feature statistics appear in the objective function of Equation 4 as the covariance matrices ψ_i. Although such flexibility can be useful, substantial simplification results by assuming the features' statistics are stationary in the image. In its strict form this assumption may be too limiting, however. This section outlines a compromise approach that was used in the implementation described in Section 6.

This method involves attaching a coordinate system to each image feature. The coordinate system has its origin at the point location of the feature, and is oriented with respect to the direction of the underlying curve at the feature point. When (stationary) statistics on feature deviations are measured, they are taken relative to these coordinate systems.

When a similar image is presented for recognition, the constant feature covariance is specialised by rotating it to orient it with respect to each image feature.

5.2 Normal Model for Feature Deviation

In the experimental domain of the previous work describing MAP model matching [6] empirical evidence was found that normal densities are a reasonable model for feature fluctuations. (The domain was matching among edge fragments from video imagery.) Although the domain of the experiments reported here is different, it is believed that normal densities are a reasonable model here as well.

6 Experimental Results

A preliminary experiment investigating the utility of the posterior marginal pose estimation approach is described in this section.

The objective function of Equation 4 was sampled in a domain of synthetic range imagery.

6.1 Preparation of Features

The features used in the experiment were oriented-range features, as described in Section 4.2. Two sets of features were prepared, the "model features", and the "image features".

The model features were derived from a synthetic range image of a truck, that was created using a modified version of the ray tracing program associated with the BRL CAD Package [11]. First range discontinuities were located in the range image by thresholding neighboring pixels, yielding range discontinuity curves. These

curves were then segmented into approximately 20-pixel-long segments via a process of line segment approximation. The line segments (each representing a fragment of a range discontinuity curve) were then converted into oriented-range features in the following manner. The X and Y coordinates of the feature were obtained from the mean of the pixel coordinates. The normal vector to the pixels was gotten via least-squares line fitting. The range to the feature was estimated by taking the mean of the pixel ranges on the near side of the discontinuity. This information was packaged into an oriented-range feature, as described in Section 4.2. The model features are shown in the first image of Figure 1. Each line segment represents one oriented-range feature, the ticks on the segments indicate the near side of the range discontinuity. There are 113 such model features.

A set of "noisy features" was generated in the following manner. The synthetic range image described above was corrupted with simulated laser radar sensor noise, using a sensor noise model that is described by Shapiro, Reinhold, and Park [12]. The resulting simulated radar range image was "restored" via a statistical restoration method of Wells and Menon [13]. The noisy features were extracted from the restored image in the manner described above for the model features. The noisy features appear in the second image of Figure 1. There are 62 noisy features. Some features have been lost due to the corruption and restoration of the range image. The set of "image features" was prepared from the noisy features by transforming the features according to a test pose, randomly deleting half of the features, and adding sufficient randomly generated features so that $\frac{1}{8}$ of the features are due to the model. The 248 image features appear in the third image of Figure 1.

6.2 Sampling The Objective Function

The objective function of Equation 4 was sampled along four straight lines passing through the location in pose space of the test pose. Oriented stationary statistics were used, as described in Section 5.1. The stationary feature covariance was estimated from a "hand match" done with a mouse between the "model features" and the "noisy features." The background rate parameter B was set to $\frac{7}{8}$.

Samples taken along a line parallel to the X axis are shown in Figure 2. The first graph shows samples taken along a 100 pixel length (the image is 256 pixels square). The second graph of Figure 2 shows samples taken along a 10 pixel length, and the third graph shows pixels taken along a 1 pixel length. The X coordinate of the test pose is 55.5, the third graph shows the peak of the objective function to be in error by about one twentieth pixel.

Samples taken along a line parallel to the μ axis are shown in Figure 3.

Each of the above graphs represents 50 equally spaced samples. The samples are joined with straight line segments for clarity. Sampling was also done parallel to the Y and ν axes with similar results.

The sampling described in this section shows that in the experimental domain the objective function has a prominent sharp peak near the correct location. Some local maxima are also apparent.

7 Related Work

Relationships between MAP model matching and previous work were described previously [6]; including work by Ayache and Faugeras [9], Goad [14], Lowe [15], Cass [16], the work of Beveridge, Weiss and Riseman [3], Hanson and Fua [17] [18], and Yuille, Geiger and Bülthoff [4].

There is a strong similarity between posterior marginal pose estimation and Hough transform (HT) methods. Roughly speaking, HT methods evaluate parameters by accumulating votes in a discrete parameter space, based on observed features. (See the survey paper by Illingworth and Kittler [19] for a discussion of Hough methods.)

In a recognition application, as described here, the HT method would evaluate a discrete pose by counting the number of feature pairings that are exactly consistent somewhere within the cell of pose space. As stated, the HT method has difficulties with noisy features. This is usually addressed by counting feature pairings that are exactly consistent somewhere nearby the cell in pose space.

The performance of the HT in the presence of noise is debatable, as discussed in [20], pp. 220. Perhaps this is due to a poor implicit noise model.

Posterior marginal pose estimation evaluates a pose by accumulating the logarithm of posterior marginal probability of the pose over image features.

The connection between HT methods and parameter evaluation via the logarithm of posterior probability has been described by Stephens [21]. Stephens proposes to call the posterior probability of parameters given image observations "The Probabilistic Hough Transform". He provided an example of estimating line parameters where image feature point probability densities were described as having uniform and normal components. The work described here differs in the underlying probabilistic model used. The present work uses the posterior probability of a pose, given an image, computed as the marginal over all possible matches to a set of model features.

8 Extensions

This paper has described a preliminary evaluation of the utility of the posterior marginal pose estimation objective function. In the domain of the experiments the objective function is seen to have a sharp peak in the proper place, and some local maxima are also apparent. An important area of further research is finding effective optimisation methods for the objective function, especially methods amenable to parallel implementation. Multiple scale search methods may prove effective here.

Applying the technique to problems of 3-D from 2-D is an important application area. One approach to the 3-D problem is the multiple-view method. Here, several instances would be run, each assuming a different view. The result from the instance achieving the best score would be chosen. This method avoids the problem of recognition-time computation of feature visibility. The

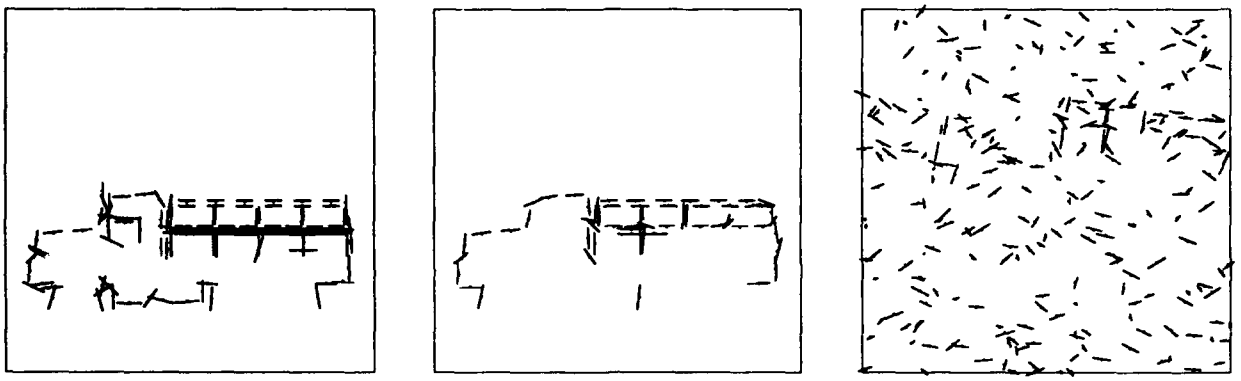


Figure 1: Model Features, Noisy Features, and Image Features

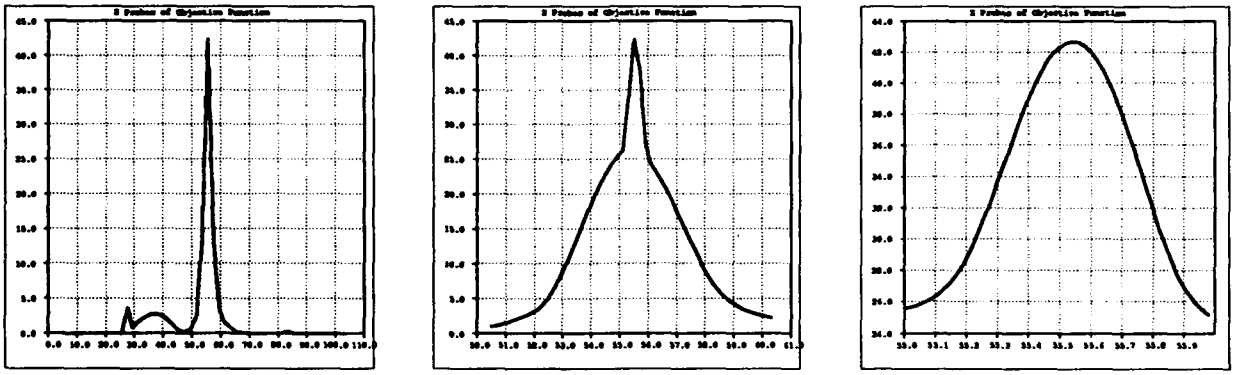


Figure 2: Objective Function Samples Along X-Oriented Line Through Test Pose, Lengths: 100 Pixels, 10 Pixels, 1 Pixel

linear projection models discussed in Section 4 can be used for specific views in 3-D orthography with scale. 3-D orthography with scale is often a good approximation of perspective projection. As an alternative to multiple views, the linear combination of models method presented by Ullman and Basri [10] could be used. A hybrid approach that might work well within a coarse-fine scheme would use the multiple view method at low resolution and linear interpolation between near views at high resolution. Another possible method would involve the conventional (non-linear) formulation of perspective projection.

More accurate objective functions would likely result from partial relaxation of the independence assumptions. One compromise approach would be Markov-style modeling of the correspondence probabilities. The probability of a feature matching some model element would be conditioned on the match state of its nearest neighbor features in the image. This would provide a better model of occlusion, since occlusion is correlated locally. Markov-style modeling of feature deviations is another likely improvement.

9 Summary

A method of estimating the pose of an object in an image, posterior marginal pose estimation, has been described. The resulting objective function was seen to have a simple form when normal feature deviation models and linear projection models are used. Several linear feature and projection models were described. Experimental results were shown indicating that in the experimental domain of synthetic range discontinuity features, the objective function has a prominent sharp peak near the correct pose. Some local maxima are were apparent. Relation to other work, and possible extensions were discussed.

10 Acknoweledgments

I thank Steve Rak and Murali Menon for their help in preparing the images used in the experiments.

References

[1] S. Barnard, "Stereo Matching by Hierarchical, Microcanonical Annealing," in *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pp. 832–835, August 1987.

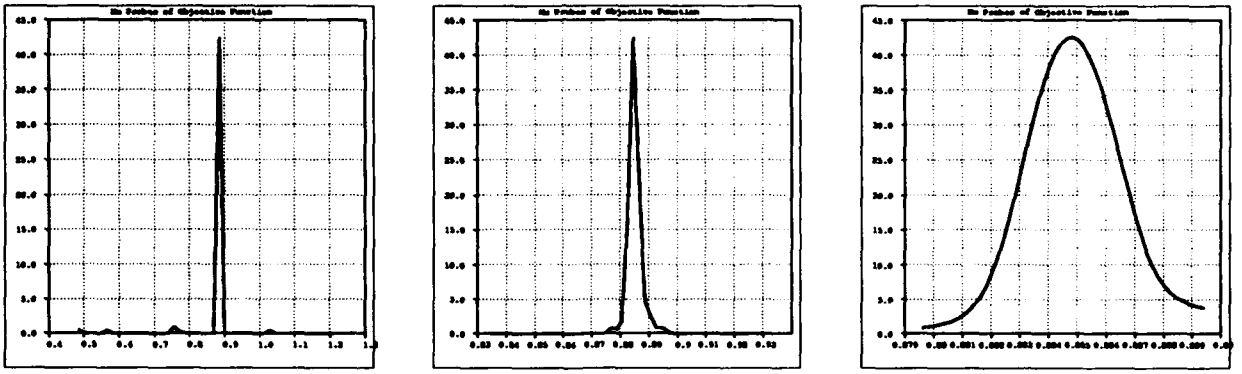


Figure 3: Objective Function Samples Along Mu-Oriented Line Through Test Pose, Lengths: .8, .1, and .01

[2] A. Blake and A. Zisserman, *Visual Reconstruction.* MIT Press, 1987.

[3] E. R. J. Beveridge, R. Weiss, "Optimisation of 2-Dimensional Model Matching," in *Proceedings: Image Understanding Workshop*, pp. 815–830, Morgan Kaufmann, 1989.

[4] A. Yuille, D. Geiger, and H. Bülthoff, "Stereo Integration, Mean Field Theory and Psychophysics," in *Computer Vision – ECCV 90*, pp. 73–82, Springer Verlag, 1990.

[5] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Transactions PAMI*, vol. PAMI-6, pp. 721–741, November 1984.

[6] W. M. Wells, "MAP Model Matching," in *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*, (Lahaina, Maui, Hawaii), pp. 486–492, IEEE, June 1991.

[7] W. Grimson and T. Lozano-Péres, "Localising Overlapping Parts by Searching the Interpretation Tree," *IEEE Transactions PAMI*, vol. PAMI-9, pp. 469–482, July 1987.

[8] J. V. Beck and K. J. Arnold, *Parameter Estimation in Science and Engineering.* John Wiley & Sons, 1977.

[9] N. Ayache and O. D. Faugeras, "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects," *IEEE Transactions PAMI*, vol. PAMI-8, pp. 44–54, January 1986.

[10] S. Ullman and R. Basri, "Recognition by Linear Combinations of Models," A.I. Memo 1152, Massachusetts Institute of Technology, August 1989.

[11] P. Dykstra and M. J. Muuss, "The BRL CAD Package: an Overview," in *Proceedings of the Fourth USENIX Computer Graphics Workshop*, (Cambridge MA), pp. 73–80, 1987.

[12] J. Shapiro, R. Reinhold, and D. Park, "Performance Analyses for Peak-Detecting Laser Radars," *Proceedings of the SPIE*, vol. 663, pp. 38–56, 1986.

[13] W. M. Wells and M. Menon, "Massively Parallel Image Restoration," in *Proceedings of the International Joint Conference on Neural Networks*, (San Diego, CA.), IEEE, 1990.

[14] C. Goad, "Fast 3-D Model Based Vision," in *From Pixels to Predicates* (A. P. Pentland, ed.), pp. 371–391, Ablex Publishing Co., 1986.

[15] D. G. Lowe, *Perceptual Organization and Visual Recognition.* Kluwer Academic Publishers, 1985.

[16] T. Cass, "A Robust Parallel Implementation of 2D Model-Based Recognition," in *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 879–884, IEEE, June 1988.

[17] P. Fua and A. J. Hanson, "Objective Functions for Feature Discrimination: Theory," in *Proceedings: Image Understanding Workshop*, pp. 443–459, Morgan Kaufmann, 1989.

[18] P. Fua and A. J. Hanson, "Objective Functions for Feature Discrimination: Applications to Semiautomated and Automated Feature Extraction," in *Proceedings: Image Understanding Workshop*, pp. 676–694, Morgan Kaufmann, 1989.

[19] J. Illingworth and J. Kittler, "A Survey of the Hough Transform," *Computer Vision, Graphics, and Image Processing*, vol. 44, pp. 87–116, 1988.

[20] W. E. L. Grimson, *Object Recognition by Computer: The Role of Geometric Constraints.* MIT Press, 1990.

[21] R. S. Stephens, "A Probabilistic Approach to the Hough Transform," in *Proceedings of the British Machine Vision Conference*, (Univ. of Oxford), pp. 55–60, September 1990.

Model-Based Object Pose in 25 Lines of Code

Daniel F. DeMenthon **Larry S. Davis**
Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3411

Abstract

In this paper we find the pose of an object from a single image when the relative geometry of four or more noncoplanar visible feature points is known. We first describe an algorithm, *POS* (Pose from Orthography and Scaling), that solves for the rotation matrix and the translation vector of the object by a linear algebra technique under the scaled orthographic projection approximation. We then describe a second algorithm, *POSIT* (POS with ITerations), that uses the pose found by POS to remove the perspective distortions from the image, then applies POS to the corrected image instead of the original image. POSIT converges to accurate pose measurements after a few cycles of image corrections and POS computations, even in conditions where perspective distortions are large. POSIT can be used with many feature points at once for added insensitivity to measurement errors and image noise. POSIT can be implemented in 25 lines or less in Mathematica; the code is provided in an Appendix.

1 Introduction

Computation of the position and orientation of an object (*object pose*) using images of feature points when the geometric configuration of the features on the object is known (a *model*) has important applications, such as calibration, cartography, tracking and object recognition. Fischler and Bolles [3] have coined the term *Perspective-n-Point problem* (P*n*P) for this type of problem with n feature points.

Researchers have formulated closed form solutions when feature points are considered in coplanar and noncoplanar configurations [1, 3, 5, 6]. However these solutions may give a false sense of safety in practice. For example the P4P problem with four coplanar points has a single solution. However, if the four points are not close to the camera, a pose that is the mirror image of the found pose with respect to a plane parallel to the image plane will project into almost the same image. Thus, with a small amount of added random error, the exact analytical solution will flip to either pose, and will have a good chance of ending with the wrong pose. An analysis which applies a scaled orthographic projection approximation is less deceiving in these situations because it will clearly show the two solutions.

Pose computations which make use of numbers of feature points larger than can be dealt with in closed form solutions are bound to be more robust because the measurement errors and image noise average out between the feature points and because the pose information content becomes highly redundant. Notable among these computations are the methods proposed by Tsai [9] and by Yuan [11] (these papers also provide good critical reviews of photogrammetric calibration techniques). The analytical expressions in these methods are independent of the number of feature points selected on the object. The methods proposed by Tsai are especially useful when the focal length of the camera, the lens distortion and the image center are not known. When these parameters have already been calibrated, the method proposed by Yuan is sufficient.

The method we describe here is related to Yuan's method in the sense that it is also based on linear algebra techniques and is used with noncoplanar points, but our approach to solving the problem is very different. We proceed in two phases.

- In the first phase we work with a scaled orthographic projection approximation (*SOP*). Finding the rotation matrix and translation vector with this assumption is dramatically simpler than with the "true" perspective projection (*TPP*) used by Yuan. We call this algorithm "POS" (Pose from Orthography and Scaling). Our equations are similar to those given by Tomasi [8] in Section 7.1.2. of his thesis on Shape and Motion from Image Streams (when he uses the shape found from previous images); one difference is that we also include the scaling of the projection and recover the three translation components of the pose. Ullman and Basri [10] use the same equations as we do (including scaling) but instead of applying them to computing the object pose, they use them to recognize new images of an object using stored images.
- In a second phase we iteratively refine the pose using a somewhat unusual approach: Since the POS algorithm requires an SOP image instead of a TPP image to produce an accurate pose, we try to syn-

thesize an SOP image from the TPP image. The approximate pose from the first phase or from the previous step allows us to "correct" the perspective image into an SOP image. We call this iterative algorithm "POSIT" (POS with ITerations). Four or five iterations are typically required to converge to an accurate pose. In this paper we characterize the performance of the algorithm by applying the algorithm in a large set of simulated situations with increasing amounts of random image perturbation [4]. In all these situations the algorithm appears to remain stable and to degrade gracefully as image noise is increased.

2 Notations

In Figure 1, we show the classic pinhole camera model, with its center of projection O, its image plane G at a distance f (the focal length) from O, its axes Ox and Oy pointing along the rows and columns of the camera sensor, and its third axis Oz pointing along the optical axis. The unit vectors for these three axes are called $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$.

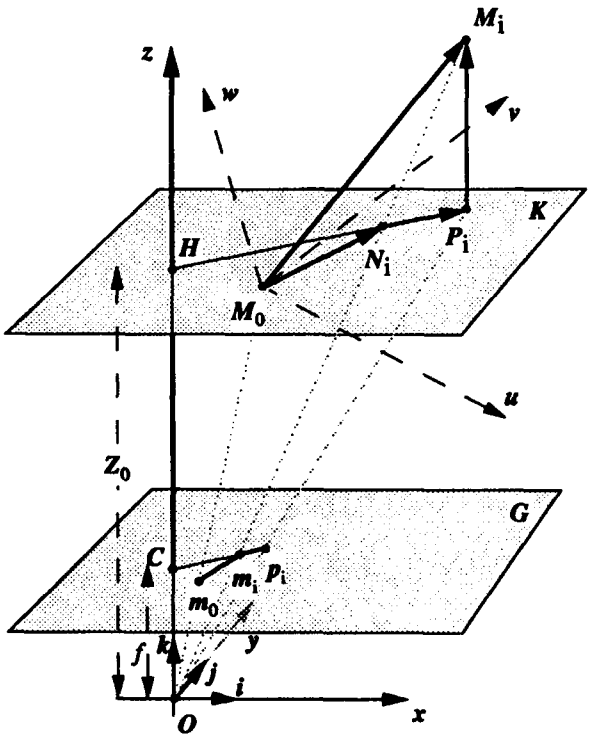


Figure 1: Perspective projection and scaled orthographic projection for an object point M_i and a reference point M_0.

An object with feature points $M_0, M_1, \ldots, M_i, \ldots, M_n$ is positioned in the field of view of the camera. The coordinate frame of reference for the object is centered at M_0 and is (M_0u, M_0v, M_0w). We call M_0 the reference point for the object. Only the object points M_0 and M_i are shown in Figure 1. The shape of the object is assumed to be known; therefore the coordinates (U_i, V_i, W_i) of the point M_i in the object coordinate frame of reference are known.

3 Scaled Orthographic Projection and Perspective Projection

3.1 Analytical Definition

Scaled orthographic projection (*SOP*) is an approximation to "true" perspective projection (*TPP*). For a given object in front of the camera, one assumes that the depths Z_i of different points M_i of the object with camera coordinates (X_i, Y_i, Z_i) are not very different from one another, and can all be set to the depth Z_0 of the reference point M_0 of the object (see Figure 1). In SOP, the image of a point M_i of an object is a point p_i of the image plane G which has coordinates

$$x'_i = fX_i/Z_0, \quad y'_i = fY_i/Z_0,$$

while for TPP an image point m_i would be obtained instead of p_i, with coordinates

$$x_i = fX_i/Z_i, \quad y_i = fY_i/Z_i$$

The ratio $s = f/Z_0$ is the *scaling factor* of the SOP. The reference point M_0 has the same image m_0 with coordinates x_0 and y_0 in SOP and TPP. The image coordinates of the SOP projection p_i can also be written as

$$x'_i = fX_0/Z_0 + f(X_i - X_0)/Z_0 = x_0 + s(X_i - X_0)$$
$$y'_i = y_0 + s(Y_i - Y_0) \qquad (1)$$

3.2 Geometric Construction of SOP

The geometric construction for obtaining the TPP image point m_i of M_i and the SOP image point m_i of M_i is shown in Figure 1. Classically, the TPP image point m_i is the intersection of the line of sight of M_i with the image plane G. In SOP, we draw a plane K through M_0 parallel to the image plane G. This plane is at a distance Z_0 from the center of projection O. The point M_i is projected on K at P_i by an orthographic projection. Then P_i is projected on the image plane G at m_i by a perspective projection. The vector m_0m_i is parallel to M_0P_i and is scaled down from M_0P_i by the scaling factor $s = f/Z_0$. Equation (1) simply expresses the proportionality between these two vectors.

4 Object Pose

4.1 Problem Definition

In a pose computation problem, we look at the inverse process: the images m_0 and m_i are given and we try to find where M_0 and M_i are required to be to create the images m_0 and m_i. In *model-based* pose computation, the relative geometric configuration of the points M_0 and M_i is known. For example, the coordinates (U_i, V_i, W_i) of M_i are given in the coordinate frame of reference (M_0u, M_0v, M_0w) of the object centered at the reference point M_0. The goal is then to find the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{T}$ of this object in the camera coordinate system.

The rotation matrix $\mathbf{R}$ for the object is the matrix whose rows are the coordinates of the unit vectors $\mathbf{i}, \mathbf{j}, \mathbf{k}$

of the camera coordinate system expressed in the object coordinate system (M_0u, M_0v, M_0w). The rotation matrix can be written as

$$\mathbf{R} = \begin{bmatrix} i_u & i_v & i_w \\ j_u & j_v & j_w \\ k_u & k_v & k_w \end{bmatrix}$$

4.2 Approximate Pose from SOP (POS)

First we find an approximate pose by assuming that the TPP image points m_i can be approximated by the SOP image points p_i (Figure 1). Our goal is to recover the coordinates of the three unit vectors $\mathbf{i}, \mathbf{j}, \mathbf{k}$ in the object coordinate system using the SOP approximation, since these coordinates are the elements of the rotation matrix. The translation vector for the object is the vector OM_0. Once we find the scaling factor of the SOP, this vector OM_0 is simply a scaled up version of the image vector Om_0. We call this pose calculation method *POS* (Pose from Orthography and Scaling).

We modify the two expressions of Equation (1). The coordinates $X_i - X_0$ and $Y_i - Y_0$ of the vector $\mathbf{M_0M_i}$ can be expressed as dot products of $\mathbf{M_0M_i}$ with unit vectors $\mathbf{i}$ and $\mathbf{j}$:

$$x_i - x_0 = s\,\mathbf{i} \cdot \mathbf{M_0M_i}, \quad y_i - x_0 = s\,\mathbf{j} \cdot \mathbf{M_0M_i}$$

We define $\mathbf{I}$ and $\mathbf{J}$ as scaled down versions of the unit vectors $\mathbf{i}$ and $\mathbf{j}$

$$\mathbf{I} = s\,\mathbf{i}, \quad \mathbf{J} = s\,\mathbf{j} \tag{2}$$

which yields

$$x_i - x_0 = \mathbf{I} \cdot \mathbf{M_0M_i}, \quad y_i - y_0 = \mathbf{J} \cdot \mathbf{M_0M_i} \tag{3}$$

The unknown scaling factor s no longer appears explicitly in Equation (3). It will be recovered as the norm of $\mathbf{I}$ or $\mathbf{J}$ once these vectors have been computed.

We express the dot products of Equation (3) in terms of vector coordinates *in the object coordinate system:*

$$\begin{aligned} [U_i\ V_i\ W_i][I_u\ I_v\ I_w]^T &= x_i - x_0, \\ [U_i\ V_i\ W_i][J_u\ J_v\ J_w]^T &= y_i - y_0 \end{aligned} \tag{4}$$

These are linear equations where the unknowns are the coordinates of vector $\mathbf{I}$ and vector $\mathbf{J}$. The other parameters are known: x_i, y_i, x_0, y_0 are the known coordinates of m_i and m_0 (images of M_i and M_0) in the camera coordinate system, and U_i, V_i, W_i are the known coordinates of the point M_i in the object coordinate system.

Writing Equation (4) for the object point M_0, for n object points $M_1, M_2, M_i, \ldots, M_n$ and their images, we generate linear systems for the coordinates of the unknown vectors $\mathbf{I}$ and $\mathbf{J}$:

$$\mathbf{A\,I} = \mathbf{x}, \quad \mathbf{A\,J} = \mathbf{y} \tag{5}$$

where $\mathbf{A}$ is the matrix of the coordinates of the object points M_i in the object coordinate system and $\mathbf{x}$ and $\mathbf{y}$ are the vectors of the x and y coordinates of the image points m_i offset by the coordinates of the image point m_0.

Generally, if we have at least three visible points other than M_0, and all these points are noncoplanar, matrix $\mathbf{A}$ has rank 3, and the solutions of the linear systems in the least square sense are given by

$$\mathbf{I} = \mathbf{B}\,\mathbf{x}, \quad \mathbf{J} = \mathbf{B}\,\mathbf{y}$$

where $\mathbf{B}$ is the pseudoinverse of the matrix $\mathbf{A}$.

We call $\mathbf{B}$ the *object matrix*. Knowing the geometric distribution of feature points M_i, we can precompute this pseudoinverse matrix $\mathbf{B}$, for example by decomposing matrix $\mathbf{A}$ by Singular Value Decomposition (SVD) [7]. This decomposition has the advantage of giving a clear diagnosis about the rank and condition of matrix $\mathbf{A}$. In [2], we discuss the geometric situations for which problems arise.

Once we have obtained least square solutions for $\mathbf{I}$ and $\mathbf{J}$, the unit vectors $\mathbf{i}$ and $\mathbf{j}$ are simply obtained by normalizing $\mathbf{I}$ and $\mathbf{J}$. As mentioned earlier, the three elements of the first row of the rotation matrix of the object are then the three coordinates of vector $\mathbf{i}$ obtained in this fashion. The three elements of the second row of the rotation matrix are the three coordinates of vector $\mathbf{j}$. The elements of the third row are the coordinates of vector k of the z-axis of the camera coordinate system and are obtained by taking the cross-product of vectors $\mathbf{i}$ and $\mathbf{j}$.

Now the translation vector of the object can be obtained. It is vector $\mathbf{OM_0}$ between the center of projection, O, and M_0, the origin of the object coordinate system. This vector, $\mathbf{OM_0}$, is aligned with vector $\mathbf{Om_0}$ and is equal to $Z_0\mathbf{Om_0}/f$, i.e. $\mathbf{Om_0}/s$. The scaling factor s is obtained by taking the norm of vector $\mathbf{I}$ or vector $\mathbf{J}$. The POS method uses at least one more point than is strictly necessary to find the object pose. At least four noncoplanar points including M_0 are required for this method, whereas three points are in principle enough if the constraints that $\mathbf{i}$ and $\mathbf{j}$ be of equal length and orthogonal are applied (see [2] for a simple pose solution for three points). Since we do not use these constraints in POS, we can verify *a posteriori* how close the vectors $\mathbf{i}$ and $\mathbf{j}$ provided by POS are to being orthogonal and of equal length. Alternatively, we can verify these properties with the vectors $\mathbf{I}$ and $\mathbf{J}$ which are proportional to $\mathbf{i}$ and $\mathbf{j}$ with the same scaling factor s. We construct a *goodness measure* G, for example as

$$G = |\mathbf{I} \cdot \mathbf{J}| + |\mathbf{I} \cdot \mathbf{I} - \mathbf{J} \cdot \mathbf{J}|$$

The goodness measure G becomes large when the results are poor and can be used for quickly testing the quality of the computed pose or for detecting wrong correspondences between image points and object points.

5 From Approximate Pose to Exact Pose: The POSIT Algorithm

5.1 Basic Idea

The POS method provides a computationally inexpensive method for directly obtaining the translation and rotation of an object; the accuracy of POS may be sufficient for tracking the motions of an object in space, or finding initial estimates for iterative methods. In this section, we present one such iterative algorithm, POSIT (POS with Iterations) , which uses POS at each iteration step. Less than five iterations are typically sufficient. The basic idea of the iteration toward the exact solution is the following:

If we could build an SOP image of the object feature points from a TPP image, we could apply the POS algorithm to this SOP image and we would obtain an exact object pose.

Computing an exact SOP image requires knowing the exact pose of the object. However, once we have applied POS to the actual image, we have an approximate depth for each feature point, and we position the feature points on the lines of sight at these depths. Then we can compute an SOP image. At the next step we apply POS to the SOP image to find an improved SOP image. Repeating these steps we converge toward an accurate SOP image and an exact pose.

5.2 Finding an SOP image from a TPP image

A TPP image point m_i (Figure 1) has coordinates

$$x_i = fX_i/Z_i, \quad y_i = fY_i/Z_i$$

and we wish to obtain the corresponding SOP image point p_i with coordinates

$$x'_i = fX_i/Z_0, \quad y'_i = fY_i/Z_0$$

Therefore the coordinates of the SOP point p_i are proportional to the coordinates of the TPP point m_i with coefficient Z_i/Z_0. In other words the SOP vector $\mathbf{Cp}_i$ is aligned with the TPP vector $\mathbf{Cm}_i$ and the proportionality factor is Z_i/Z_0:

$$\mathbf{Cp}_i = \frac{Z_i}{Z_0}\mathbf{Cm}_i \tag{6}$$

5.3 Finding an SOP image in POSIT

After applying POS once using the TPP image, we do not have the actual Z_i for each point M_i, but we can compute the approximation

$$Z_i = Z_0 + \mathbf{M_0M}_i \cdot \mathbf{k} \tag{7}$$

where $\boldsymbol{k}$ is the unit vector along the optical axis Oz. Expressed in the object coordinate system, $\boldsymbol{k}$ is the third row of the rotation matrix, and $\mathbf{M_0M}_i$ is a known vector with coordinates (U_i, V_i, W_i). Thus Equation (7) becomes

$$Z_i = Z_0 + k_u U_i + k_v V_i + k_w W_i \tag{8}$$

and the SOP point p_i is obtained by combining Equations (6) and (8):

$$\mathbf{Cp}_i = (1 + \frac{s}{f}(k_u U_i + k_v V_i + k_w W_i))\ \mathbf{Cm}_i \tag{9}$$

where we have replaced $1/Z_0$ by s/f, the ratio of the scaling factor found in POS by the camera focal length. Expression (9) provides at each step of the POSIT algorithm the approximated positions of the SOP image points p_i in relation to the image points m_i using the third row of the computed rotation matrix.

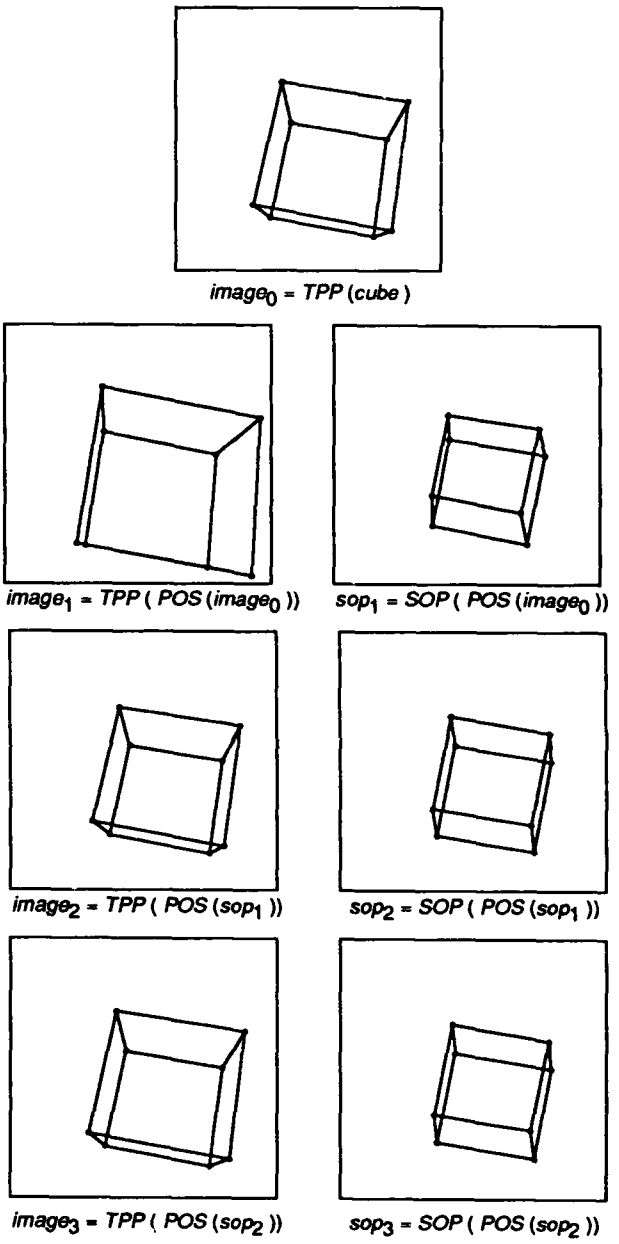


Figure 2: TPP images (left) and SOP images (right) of the cube poses computed at successive steps by the POSIT algorithm.

6 Illustration of the Iteration Process in POSIT

To illustrate the iteration process of POSIT, we apply the method to synthetic data. The object is a cube of size 10 cm. The cube is assumed transparent, and the points of interest are the corners of the cube. We use a full cube in this experiment without simulating hidden faces because it is interesting to see the converging projections of the parallel edges in the TPP image being transformed into parallel projections in the SOP image (in fact it is not difficult to do actual experiments with eight corners of a cube, using light emitting diodes mounted in a cubic arrangement on a transparent frame). The image size is assumed to be 512 × 512 pixels, and the focal length is 760 pixels, providing an angular field of view of 37 degrees with the assumed image size. The corners of the cubes are at a distance between three and four times the cube size from the center of projection of the camera. The projection on top of Figure 2 is the given image for the cube. We can write this operation as

$$image_0 = TPP(cube).$$

The projections of the cube edges are shown, although they are not used by the algorithm. The enclosing square is the boundary of the 512 × 512 pixel image area. Because the distance-to-size ratio for the cube is small, the cube image shows strong convergence of image edges for the cube edges almost parallel to the camera optical axis. With such an image the POS algorithm does not give a good approximation of the rotation and translation of the cube. One can have an idea of the success of the POS algorithm by computing a TPP image of the cube at the found pose. The three projections of the left column in Figure 2 are such TPP projections at successive iterations. The transformation which gave the top left projection can be written as

$$image_1 = TPP(POS(image_0)$$

where $image_0$ is the given image and $POS()$ represents the inverse perspective operation found by the POS algorithm. As can be seen from this projection, the world points found by POS were not very good. Note that POSIT does not compute $image_1$. Instead, POSIT computes an SOP image using only the actual image corners and the depths it computed for the corners. This image is shown on the top right in Figure 2. Equation (9) is used. This operation is written as

$$sop_1 = SOP(POS(image_0))$$

Notice that in the resulting projection the SOP images of parallel edges of the cube are not yet quite parallel. The reason is that the found corner depths are still approximations. The next iteration of POSIT uses this SOP image, and at this step it finds a shape closer to the cube, as illustrated by the fact that its perspective projection $image_2$ (again, this image is not actually computed by the algorithm) is very close to the original image. The operation for finding $image_2$ is

$$image_2 = TPP(POS(sop_1))$$

The next SOP image found by POSIT has parallel edges. This image is

$$sop_2 = SOP(POS(sop_1))$$

The next iteration, illustrated at the bottom of Figure 2, brings only minor improvements.

7 Protocol of Performance Characterization

In this section, we try to follow the recommendations of Haralick for performance evaluation in computer vision [4]. We compute the orientation and position errors of the POS and POSIT algorithms for two objects at ten distances from the camera with 40 random orientations for each distance. Synthetic images are created with three levels of image noise for each combination, and the poses of the object computed by POS and POSIT from these images are compared with the actual poses.

7.1 Objects

The two objects are

1. A configuration of four points from a 10 cm cube, one corner taken as reference point, and its three adjacent corners. The four points form a tetrahedron (Figure 3, top).
2. The eight corners of a 10 cm cube. The reference point is one of the corners (Figure 3, bottom).

The size of both objects is taken to be 10 cm and is used to measure the displacement of the objects with respect to the camera. Note however that the maximum distance between two points in the tetrahedron configuration is also 10 cm, whereas the maximum distance for the cube is 17.3 cm, between two diagonal corners. Thus the cube image may be larger than the tetrahedron in some orientations at the same distance from the camera, and be more distorted by perspective.

7.2 Object Positions

The reference points of the objects are positioned on the optical axis at ten distances from the center of projection, from 40 cm to 400 cm, i.e. from four times to 40 times the size of the objects (Figure 3). These distance-to-size ratios are used as the horizontal coordinates on all the orientation and position error plots.

Around each of these reference point positions, the objects are rotated into 40 random orientations. The rotation matrices defining these 40 orientations are computed from three Euler angles chosen by a random number generator in the range $(0, 2\pi)$.

7.3 Image Generation

We obtain images by perspective projection with a focal length of 760 pixels. Here we do not clip the image, in order to allow for large offsets of the images. When the reference point of the cube is 40 cm from the image plane on the optical axis and when the cube is completely on one side of the optical axis, the point at the diagonal of the cube may be 30 cm from the image plane and have an image 355 pixels from the image center. Only

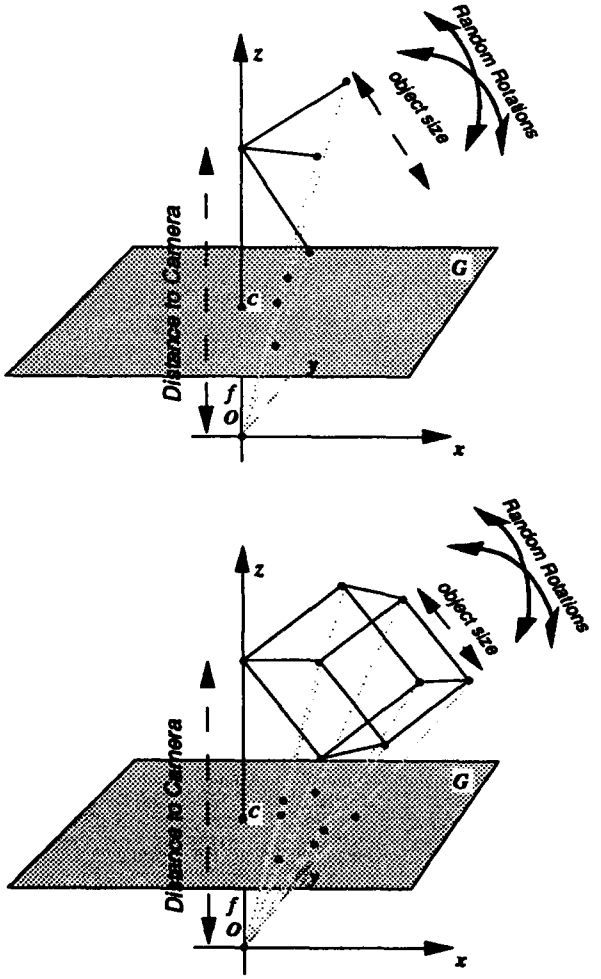


Figure 3: Object and parameter definitions for rotation and translation error estimations. *Top*: Tetrahedron. *Bottom*: Cube.

a wide-angle camera with a total angular field of more than 50 degrees would be able to see the whole cube in this position.

We specify three levels of random perturbation and noise in the image. At noise level 1, the real numbers computed for the coordinates of the perspective projections are rounded to integer pixel positions. At noise level 2, the integer positions of the lowest level are disturbed by vertical and horizontal perturbations of $\pm$ 1 pixel around the integer positions. These are created by a uniform random number generator. At noise level 3, the amplitude of the perturbations are $\pm$ 2 pixels. When the objects are at 400 cm from the camera, the image may be as small as 20 pixels, and a perturbation of two pixels on each side of the image produces a 20% perturbation in image size.

7.4 Orientation and Position Errors

For each of the images, the orientation and position of the object are computed by the POS algorithm, then by the POSIT algorithm for four more iterations (a total of five steps including the initial POS step). These orientations and positions are compared to the actual orientation and position of the object used to obtain the image. We compute the axis of the rotation required to align the coordinate system of the object in its actual orientation with the coordinate system of the object in its computed orientation. The *orientation error* is defined as the rotation angle in degrees around this axis required to achieve this alignment. Details of this computation are given in [2]. The *position error* is defined as the norm of the translation vector required to align the computed reference point position with the actual reference point, divided by the distance of the actual reference point position from the camera. Thus the position error is a relative error, whereas the orientation error is a measure in degrees.

7.5 Combining the Results of Multiple Experiments

As mentioned above, for each distance-to-size ratio, many rotations are considered. We compute the average and standard deviation of the orientation and position errors over all these rotations and plot the averages with their standard deviation error bars as a function of the distance-to-size ratios. Each plot shows the results both for POS, and for POSIT after five iterations. The plots for the orientation error are shown in Figure 4 , and the plots for the position errors are shown in Figure 5. In each of these two figures, the plots in the left column are for the tetrahedron, and the plots in the right column are for the cube. The top diagrams are for the lowest image noise level, the middle diagrams for the medium noise level, and the bottom diagrams for the highest noise level.

8 Analysis of the Pose Error Diagrams

8.1 Comparison between POS and POSIT

POSIT provides dramatic improvements over POS when the objects are very close to the camera, and almost no improvements when the objects are far from the camera. When the objects are close to the camera, the so-called *perspective distortions* are large, and the approximation that the image is an SOP is poor; therefore the performance of POS is poor. For the object reference points at the shortest distance-to-size ratio (4), errors in orientation evaluation are in the 10° region, and errors in position evaluation are in the 10% region. When the objects are very far, there is almost no difference between SOP and TPP; thus POS gives the best possible results, and POSIT cannot improve upon them. POS gives its best performance for distances around 30 times the object size for low image noise, and around 20 times for high image noise, with orientation errors in the 3° region and position level in the 3% region. At very low to medium range and low to medium noise, POSIT gives poses with less than 2° rotation errors and less than 2% position errors.

8.2 Variation of Noise Effects with Range

At short range, the images of the objects are large and perturbations of a few pixels are small compared to the

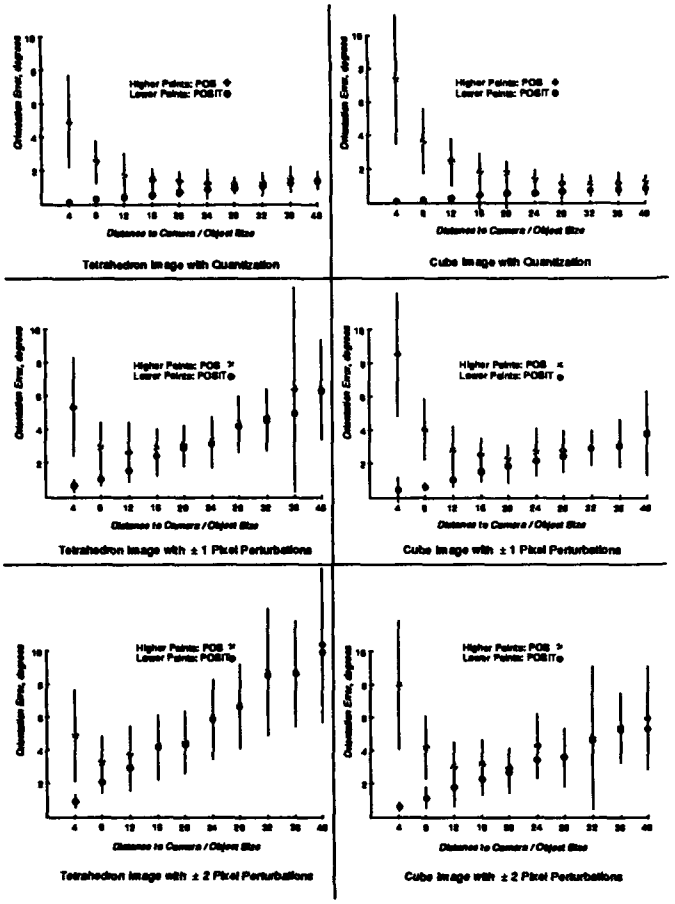


Figure 4: Angular orientation errors at various distances for a tetrahedron (left) and for a cube (right) at three image noise levels (quantization, ±1 pixel, ±2 pixels).

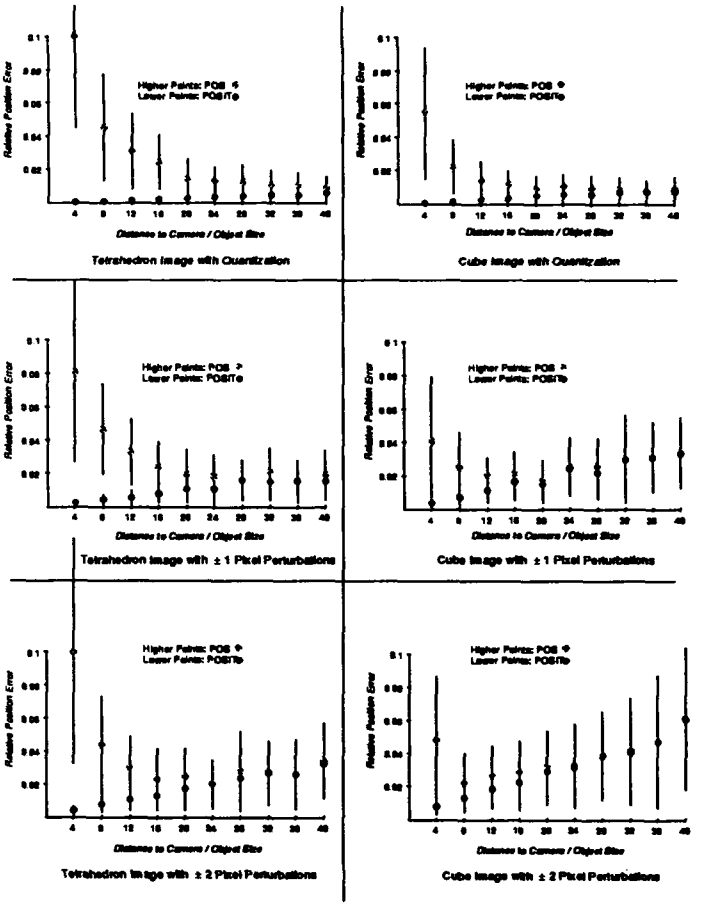


Figure 5: Relative position errors at various distances for a tetrahedron (left) and for a cube (right) at three image noise levels (quantization, ±1 pixel, ±2 pixels).

total size of the image. At long range the images occupies only around 20 pixels and perturbations of a few pixels may change the relative distribution of the image points. Therefore in the diagrams with larger noise levels, the pose errors of the algorithms increase as the distance ratios increase from 30 to 40.

8.3 Comparison between Cube and Tetrahedron

The long error bars at short range for POS are due to the fact that the apparent image size can be very different depending on the orientation. For example, the cube looks like an object of size 10 cm when a face is parallel to the image plane, but one dimension is 70% larger when a cube diagonal is parallel to the image plane. In this last configuration, the reference point projects at the image center whereas the opposite corner is offcentered by more than 323 pixels, with a large resulting perspective distortion. The tetrahedron does not have as large apparent size changes, which explains the fact that at short viewing distance the average error and standard deviation produced by POS are smaller for this shape than for the cube. This is more an artifact of the problem of defining object size with a single number than a specific advantage of the tetrahedron over the cube.

At high noise level and long range, the performance with the cube becomes almost twice as good as with the tetrahedron for POS and POSIT, because the least square method averages out the random errors on the points, and the averaging improves when more points are made available to the method.

9 Convergence Analysis

With the distance-to-size ratios used in the rotation and translation error evaluations above, the POSIT algorithm would converge in four or five iterations. The convergence test in the POSIT algorithm consists of quantizing (in pixels) the coordinates of the image points in the SOP images obtained at successive steps, and terminating when two successive SOP images are identical (see Appendix A).

Calculations with 1D images of a 2D world show that the quantities determining the algorithm convergence are ratios of image coordinates over focal length. When all the ratios are smaller than 1, the algorithm converges. Therefore with a camera with a 90 degree total field of view, the algorithm would converge with all possible image points. When these ratios are larger than 1 for all the coordinates, the algorithm diverges. Thus with an

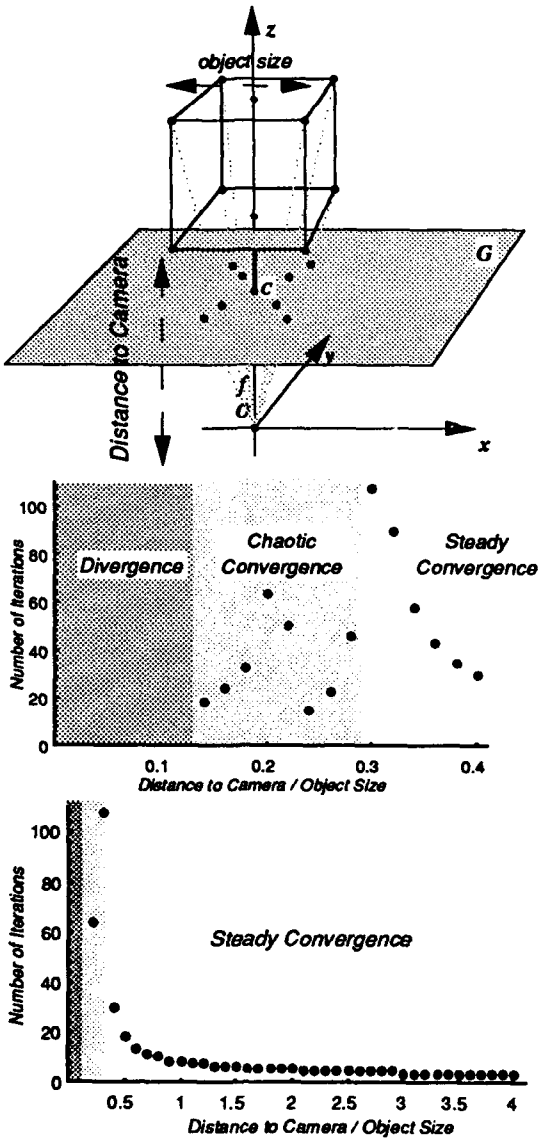


Figure 6: Number of iterations for POSIT as a function of distance to camera. *Top*: Definition of distance and object size. *Middle*: Convergence analysis at very close ranges. Convergence occurs if the 10 cm cube is more than 1.2 cm away from the camera. *Bottom*: Number of iterations for a wider range of distances.

object with all its image points at the periphery of the field of a 110 degree camera the algorithm would diverge. In mixed situations with small and large ratios, mixed results are obtained as to whether the small ratios win convergence over the large ratios or not. Therefore it is still possible to obtain convergence with image points at the periphery of a very wide-angle field if some of the image points are close to the center of the field.

Simulations appear to show similar properties with 2D images in a 3D world. In these simulations, a cube is displaced along the camera optical axis (Figure 6). One face is kept parallel to the image plane, because at the shorter ranges being considered, the cube cannot be rotated much without intersecting the image plane. The distance used to calculate the distance-to-object size ratio in the plots is the distance from the center of projection to the cube. Noise of $\pm$ 2 pixels is added to the perspective projection. For a cube of 10 cm, four iterations are required for convergence until the cube is 30 cm from the center of projection. The number gradually climbs to eight iterations as the cube reaches 10 cm from the center of projection, and 20 iterations for 5 cm. Then the number increases sharply to 100 iterations for a distance of 2.8 cm from the center of projection. In reference to our prior 1D observations, at this position the images of the close corners are more than two focal lengths away from the image center, but the images of the far corners are only half a focal length away from the image center and probably contribute to preserving the convergence.

Up to this point the convergence is monotonic. At still closer ranges the mode of convergence changes to a nonmonotonic mode, in which SOP images of successive steps seem subjected to somewhat random variations from step to step until they hit close to the final result and converge rapidly. The number of iterations ranges from 20 to 60 in this mode, i.e. less than for the worse monotonic case, with very different results for small variations of object distance. We label this mode "chaotic convergence" in Figure 6. Finally, when the cube gets closer than 1.2 cm from the center of projection, the differences between images increase rapidly and the algorithm clearly diverges. Note, however, that in order to see the close corners of the cube at this range a camera would require a total field of more than 150 degrees, i.e. a focal length of less than 1.5 mm for a 10 mm CCD chip, an improbable configuration. This preliminary convergence analysis and the experiments of the previous section show that the POSIT algorithm converges in a few iterations for most camera and object configurations.

10 Conclusions

We have presented an algorithm, POSIT, that can compute the pose of an object using several noncoplanar feature points of the object at a low computational cost. The algorithm consists of computing an approximate pose by a method (POS) based on scaled orthographic projection and a least square technique, and replacing the original image with an image obtained by scaled orthographic projection (SOP) using the approximate depths of the feature points given by this pose. Then POS can be applied to this image to produce a more accurate object pose. This SOP-POS cycle is repeated until the improvements become smaller than the precision required by the application. Convergence tests show that the algorithm can be made to diverge only when all the image points are taken at the edges of a very wide field of view. Very few iteration steps are typically required to converge to an accurate pose. We have characterized the performance of the POS and POSIT algorithms by applying them to a large set of simulated situations with increasing amounts of random image perturbation. The

POSIT algorithm appears to remain stable and to degrade gracefully with increasing image noise levels. Applications of the POSIT algorithm include camera calibration, real-time target tracking and object recognition.

Acknowledgements

We would like to thank Thor Bestul for enlightening discussions. The support of the Defense Advanced Research Agency under Contract DACA76-88-C-0008 is gratefully acknowledged.

Appendix A: A Mathematica program implementing POS and POSIT

```
(* Compute the pose of an object given a list of 2D image points *)
(* a list of corresponding 3D object points *)
(* and the pseudoinverse matrix for the list of object points *)
(* The output has two elements, the pose computed by POS *)
(* and the pose computed by POSIT *)
(* once the SOP image points don't move from one step to the next *)
(* The first point of the object point list is taken as reference point M0 *)
(* Copyright Daniel DeMenthon 1991 *)


POSIT[imagePoints_,objectPoints_,objectMatrix_,focalLength_]:= Module[
{objectVectors, imageVectors, IVect, JVect, ISquare, JSquare, IJ, imageDifference,
row1, row2, row3, scale1, scale2, scale, oldSOPImagePoints, SOPImagePoints,
 translation, rotation, firstPose, count=0, converged = False},
objectVectors = (#-objectPoints[[1]])& /@ objectPoints;
oldSOPImagePoints=imagePoints;
(* loop until difference between 2 SOP images is less than one pixel *)
While[! converged,
  If[count==0,
    (* we get image vectors from image of reference point for POS: *)
    imageVectors = (# - imagePoints[[1]])& /@ imagePoints,
    (* else count>0, we compute a SOP image first for POSIT: *)
     SOPImagePoints = imagePoints (1 + (objectVectors.row3)/translation[[3]]);
     imageDifference = Apply[Plus, Abs[Round[Flatten[SOPImagePoints]]-
                                Round[Flatten[oldSOPImagePoints]]]];
    oldSOPImagePoints = SOPImagePoints;
   imageVectors = (# - SOPImagePoints[[1]])& /@ SOPImagePoints
 ]; (* end else count>0*)
 {IVect, JVect} = Transpose[objectMatrix . imageVectors];
 ISquare = IVect.IVect; JSquare = JVect.JVect; IJ = IVect.JVect;
 {scale1, scale2} = Sqrt[{ISquare, JSquare}];
 {row1, row2} = {IVect/scale1, JVect/scale2};
 row3 = RotateLeft[row1] RotateRight[row2] -RotateLeft[row2] RotateRight[row1];
 rotation={row1, row2, row3};
 scale = (scale1 + scale2)/2.0; (* scaling factor in SOP *)
 translation = Append[imagePoints[[1]], focalLength]/scale;
 If[count==0, firstPose = {rotation, translation}];
 converged = (count>0) && (imageDifference<1);
 count++
]; (* End While *)
Return[{firstPose,{rotation, translation}}]]
```

Example of input

```
fLength= 760;

cube={{0,0,0},{10,0,0},{10,10,0},{0,10,0},{0,0,10},
   {10,0,10},{10,10,10},{0,10,10}};

cubeMatrix = PseudoInverse[cube];

cubeImage= {{0,0},{80,-93},{245,-77},{185,32},{32,135},
      {99,35},{247, 62},{195, 179}};

{{POSRot,POSTrans},{POSITRot,POSITTrans}} =
          POSIT[cubeImage, cube, cubeMatrix, fLength];
```

References

[1] D. DeMenthon and L.S. Davis, "New Exact and Approximate Solutions of the Three-Point Perspective Problem", 1990 International Conference on Robotics and Automation, Cincinnati, pp. 40–45.

[2] D. DeMenthon and L.S. Davis, "Model-Based Object Pose in 25 Lines of Code", University of Maryland, Center for Automation Research Technical Report, in preparation.

[3] M.A. Fischler and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", Comm. ACM, vol. 24, pp. 381–395, 1981.

[4] R.M. Haralick, "Performance Characterization in Computer Vision", University of Washington C.S. Technical Report, July 1991.

[5] R. Horaud, B. Conio and O. Leboulleux, "An Analytical Solution for the Perspective-4-Point Problem", Computer Vision, Graphics, and Image Processing, vol. 47, pp. 33–44, 1989.

[6] S. Linnainmaa, D. Harwood, and L.S. Davis, "Pose Determination of a Three-Dimensional Object using Triangle Pairs", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 10, pp. 634-647, 1988.

[7] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Veterling, Numerical Recipes in C, Cambridge University Press, Cambridge, UK, 1988.

[8] Carlo Tomasi, "Shape and Motion from Image Streams: A Factorization Method", Technical Report CMU-CS-91-172, Carnegie Mellon University, September 1991.

[9] R.Y. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," IEEE J. Robotics and Automation, vol. 3, pp. 323–344, 1987.

[10] S. Ullman and R. Basri, "Recognition by Linear Combinations of Models", A.I. Memo 1152, MIT AI Laboratory, August 1989.

[11] J.S.C. Yuan, "A General Photogrammetric Method for Determining Object Position and Orientation", IEEE Trans. on Robotics and Automation, vol. 5, pp. 129–142, 1989.

Representation Issues in Bayesian Decision Theory for Planning and Active Perception

Thomas Dean* **Jak Kirman**
Department of Computer Science
Brown University, Box 1910, Providence, RI 02912

Abstract

Bayesian methods provide a theoretically pleasing approach for optimal decision making. However, while it is often easy to specify a model for a given decision problem, it is much more difficult to gather the necessary statistics to quantify the model and implement a decision procedure with reasonable response time. In robotics applications, decision models are complicated by having to deal with the temporal, spatial, and perceptual aspects of the problem. We examine a number of representational techniques, exploring the issues that arise with regard to quantification and efficient implementation. In particular, we consider discrete and continuous temporal representations, volumetric and location-based spatial representations, and time-separable and finite-horizon performance measures. The techniques considered provide the designer of robot decision procedures with alternatives in coping with the complexity that arises in realistic problems; alternatives that can be justified in terms of tradeoffs between modeling accuracy and computational complexity.

Introduction

Bayesian techniques have been developed for sensor fusion [Moravec and Elfes, 1985], value of information calculations [Dean and Wellman, 1991], state estimation [Durrant-Whyte, 1988, Hager, 1990], and motion planning under uncertainty. However, in focusing on component problems, computational and knowledge acquisition issues are often ignored or at least discounted. In this paper, we explore some of the issues that arise in quantifying (gathering statistics for) Bayesian decision models and designing efficient inference algorithms.

*This work was supported in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601, by the Advanced Research Projects Agency of the Department of Defense monitored by the Air Force under Contract No. F30602-91-C-0041, and by the National Science foundation in conjunction with the Advanced Research Projects Agency of the Department of Defense under Contract No. IRI-8905436.

Preliminaries

A *probabilistic network* is represented as a directed graph $G = (V, E)$. A vertex v is said to be an *immediate predecessor* of a vertex v' just in case there is an edge from v to v' in E. The vertices in V correspond to random variables called *chance nodes*. The edges in E define the causal and informational dependencies between the random variables. Chance nodes are discrete or real valued variables that encode states of knowledge about the world. Let Ω_C be the set of values of a chance node C. There is a probability distribution $\Pr(C = \omega, \omega \in \Omega_C)$ for each node. If the chance node has no predecessors then this is its marginal probability distribution; otherwise, it is a conditional probability distribution dependent on the states of the immediate predecessors of C in G. Pearl [1988] provides a comprehensive treatment of probabilistic networks.

In decision making tasks, nodes in the graph are often *instantiated* by fixing the value of the corresponding random variable. Instantiation might correspond to obtaining evidence or performing some sort of hypothetical reasoning. The primary operation on probabilistic networks that we are concerned with involves computing the posterior distribution of each node given the instantiated nodes.

Markov processes provide a simple, well-understood method of representing change over time. To represent Markov processes, we introduce a special case of probabilistic networks [Dean and Kanazawa, 1989]. Given a set of *state* variables, $\mathcal{X}$, and a finite ordered set of states or time points, $\mathcal{T}$, we construct a set of chance nodes, $\mathcal{C} = \mathcal{X} \times \mathcal{T}$, where each element of $\mathcal{C}$ corresponds to the value of some particular $x \in \mathcal{X}$ at some $t \in \mathcal{T}$. Let $\mathcal{C}_t$ correspond to the subset of $\mathcal{C}$ restricted to t. A process model is said to be k-Markov just in case,

$$\Pr(C_t | C_{t-1}, C_{t-2}, \ldots) = \Pr(C_t | C_{t-1}, \ldots, C_{t-k}).$$

For decision purposes, we are generally interested in the outcomes resulting from particular sequences of actions. To represent action sequences, we add state variables corresponding to the actions taken in each state. Outcomes correspond to possible instantiations of the

state variables. We employ a *value function* to determine the value of outcomes. In the case of probabilistic outcomes, we use the expectation of value over all outcomes as a measure to guide decision making. To evaluate a particular action sequence, we instantiate the action state variables using the actions in the given sequence, compute the posterior distribution given the instantiated variables, and determine the expected value using the value function and the posterior distribution.

In general, the value function ranges over the product of the set of values for all of the random variables, complicating decision making by requiring that we enumerate this very large space of outcomes. In practice, we often employ a simpler, *time-separable* value function. By time separable, we mean that the total value is a (perhaps weighted) sum of the value at the different time points. If V_t is the value function at time t, then the total value, V, is defined as

$$V = \sum_{t \in \mathcal{T}} \gamma(t) V_t,$$

where $\gamma : \mathcal{T} \rightarrow \{x | 0 \leq x \leq 1\}$ is a function of time used to weight future consequences.

To represent variable intervals of time separating states, we encode duration information as real-valued state variables. Other continuous variables can be introduced to handle resources like fuel or rations, or factors like precision or machine downtime. As a consequence of our employing continuous chance variables, we are forced to employ sampling methods [Peot and Shachter, 1991] as the principal method of estimating the posterior distribution for our networks.

Robot Decision Making

In the mobile target localization (MTL) problem, a robot is given the task of tracking and periodically reporting on the position of a mobile target. We represent the dynamics in terms of a Markov chain with state variables corresponding to the location of the robot, the location of the target, the observations of the robot, the movement action, and the report action. Movement actions correspond to invoking particular navigation procedures. In order that the interval of time separating successive time points be the same, we assume that the navigation procedures run for a set amount of time. This reliance on fixed duration actions is somewhat unnatural and complicates the process of quantifying the model. Performance is measured in terms of a time-separable value function for which the value at time t is the accuracy of the report issued at t.

In the Fixed Target Reconnaissance (FTR) problem, the robot is given a task involving covert search and survey; the robot is sent out to find and report on the location of some set of fixed targets, while at the same time avoiding detection by some set of either fixed or mobile observers. Avoiding detection involves preventing line-of-sight contact with observers. Potential threats of detection might be determined beforehand in the case of known intervals coinciding with satellites being positioned overhead or ground-based observers on periodic inspection tours. Alternatively, the robot might employ fixed-location sensors or deploy relocatable sensors to detect and supply advance warning with regard to less predictable observers.

The FTR problem introduces some interesting representation issues. For instance, it is more difficult to accept that all actions take the same amount of time. Other issues involve tradeoffs regarding detection, target localization, and the time at which the robot reports on the location of the target.

Fixed Target Reconnaissance

In order to illustrate some of the issues involved in representing robot decision problems, we consider the FTR problem in more detail. First, we present a probabilistic network that represents the FTR problem assuming that all the actions take the same amount of time; we provide a time-separable value model for a particular instance of the FTR problem. Second, we look at the issues that arise when we relax the restriction on actions taking a fixed amount of time; we provide another value model that approximates our notion of performance in this case and examine the conditions under which this approximation is reasonable. We then look at some of the issues involved in incorporating evidence gathered from the environment, paying attention to the fact that the computational expense of evaluating these probabilistic networks grows too quickly if they are used to represent all the information available to us; we suggest an approach that allows us to take much of the information into account without severely degrading performance.

Modeling the FTR Decision Problem

Figure 1 shows a probabilistic network representing the FTR problem. Shaded nodes represent direct measurements. Square nodes represent actions, diamond-shaped nodes represent value nodes used to encode a performance measure for the underlying decision problem, and unshaded circular nodes represent states of the world that cannot be observed directly.[1]

In each time slice, there are 8 state variables. O_r ranges over observations made about the robot's location, O_t over the observations made about the target's location. L_r ranges over the possible locations of the robot, and L_t over the locations of the target. D is a boolean variable indicating whether or not the robot is detected. A ranges over possible actions the robot will take next, and V is a real-valued variable. O_e ranges

[1] We employ graphical conventions typically used in displaying *influence diagrams* [Howard and Matheson, 1984], a generalization of probabilistic networks.

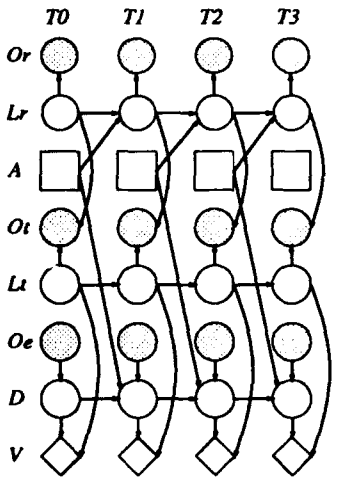


Figure 1: FTR model with constant time intervals

over the observations made by the robot regarding observers wishing to detect the robot's presence.

In the network shown in Figure 1, looking at the time slices after the first, the arc into node O_r indicates that the observations made about the robot's whereabouts depend on where it is. The arcs into node L_r indicate that the location of the robot depends on what action it took previously and where it was one time step ago. Those entering node O_t indicate that the observations made about the location of the target depend on where the target is and where the robot is. Those entering node L_t indicate that the location of the target depends on where it was before (it does not move). Finally, the arcs into node D specify that whether the robot has been detected depends on whether it was detected before, what action it took (one of the actions may be to hide), and whether there appear to be any observers around. The value node indicates the value of a situation described in terms of the target's location and whether or not the robot is detected.

The topology of this network represents the problem at a relatively high level — the elements of interest to us, such as the location of the target and the robot, whether the robot has been detected etc. By changing the spaces over which the random variables range, the same network can be used for robots moving inside buildings, in an urban environment, or in the countryside. The quantification of the network — the specification of the conditional probability tables for each node — will of course vary depending on the spaces involved. In the following, suppose that the network is being used to direct the search for an object within a room, or small number of rooms; the actions allow the robot to move from one part of a room to another, to extract and interpret camera images, and to gather and analyze sonar data.

Maximizing Expected Value

The general description of the problem provided so far is not sufficient to unambiguously specify a performance measure. In the following, we consider a more detailed specification of the problem, and examine how our specification affects our representation of performance.

Suppose that the robot is part of an advance unit that has infiltrated an enemy installation in which it is known that there is a missile silo. A tank unit is approaching the area, and the robot has some distribution, $\Pr(Q = t)$, for estimating its time of arrival. The robot must locate the silo, and report the location to the tank unit when it arrives, at time Q (Query). The only penalty associated with being discovered is that the robot will not be able to carry on its search of the area. There is a positive reward V_g (good) for giving the correct answer when the tank unit arrives, and a negative reward V_b (bad) for giving an incorrect answer.

We measure performance for this version of the problem as follows. The value of a sequence of actions is the sum of the values for each time slice, weighted by the likelihood that the tank unit will arrive between that time slice and the next. The value of a sequence of actions for a particular time is the expected reward for giving the best response we can at that time. Since actions enable observations, we have to account for the value of the information gained through observation. The probability of giving a correct response for a particular observation given an action sequence is

$$PC_t(o|s) = \max_{u \in \Omega_{L_t}} \Pr(\langle L_t = u, t\rangle | \langle O = o, t\rangle, s),$$

where $o \in \Omega_o$ and s is an action sequence.[2] The expected value of choosing the correct location if we perform this action sequence and make this observation is

$$V_t(o|s) = V_g PC_t(o|s) + V_b(1 - PC_t(o|s)).$$

We assume that the actions and observations are independent of the time at which we are asked to report on the location of the target.

We define the time-separable value function capturing the expected value of executing the sequence s if we are requested to report the location of the target at time t as

$$V_t(s) = \sum_{o \in \omega_o} \Pr(\langle O = o, t\rangle | s) V_t(o|s).$$

The total value V is the value of performing the sequence at each of the next n steps in time, weighted by the likelihood that it is at that time step that we are asked to make the report,

$$V(s) = \sum_{t \in \mathcal{T}} \Pr(t < Q \leq t + 1) V_t(s).$$

[2] $\langle \mathcal{X} = x, t\rangle$ is read as "state variable $\mathcal{X}$ has value x at time t".

This sum does not account for there being a finite horison, but this omission can be easily remedied by adding another term for the value of the last location estimate weighted by the probability that the request for a report comes after the last time point accounted for in the model, $\Pr(Q > \max(t))V_{\max(t)}(s)$. Because the above value function is time-separable, the computation is substantially less expensive than it would be otherwise.

Notice that we have made several simplifying assumptions in order to obtain a time-separable function. The most important of these is that, given the current observations and those at some time t in the future, we can estimate the distributions for the nodes at time t without enumerating all possible sequences of observations *between* now and t. Another major assumption we have made is that the time between slices is constant and independent of the action taken. This assumption is not usually reasonable; different actions take different amounts of time, and, more importantly, the same action may take a different amount of time depending on the circumstances in which it is invoked.

In order to take variable length actions into account, we introduce an extra, continuous-valued, node at each time slice, representing the total elapsed time since the beginning of execution. The network in Figure 2 includes these nodes. Now, whether or not we are detected depends on the duration of the previous action (the difference between the time of the start and the time of the end of the action), and the time an action finishes depends on the time it starts and what action it was. Here, the time is an observation we can make directly; once an action is completed, we know what the current time is. Notice that in general, the time will influence states of the world. In this particular case, the location of the target is fixed, so it does not vary with time, and we assume that the location of the robot is independent of time given the action performed.

The value function in this case is not much more complicated than it was before; the important point to note is that the time is one of the observations, and for each possible individual observation, we can compute the value at that time, weighted by the likelihood of being requested to report the location of the target at that time. So the expression $\Pr(Q = t)$ needs to be factored into the computation as

$$V_t(o|s) = \Pr(T_t < Q < T_{t+1}) \\ ((V_g PC_t(o|s) + V_b(1 - EV_t(o|s)))).$$

Note that since T_t is a part of the observations, it is fixed at this point. Notice also the distinction between the subscript t, which denotes the tth time slice, and the value of T_t, which is the total time elapsed since the beginning of the execution. For different values of o and the same value of t, T_t may be different. The probability that the query will be made between time slice t and time slice $t + 1$ clearly depends on the duration of the action made at time t. In general, this term must be computed as an average of the possible durations of the action; this is correct only if the expected value of the situation resulting from the action is independent of the duration of the action. Although this is unlikely to be true, it is a reasonable approximation in the case where actions take a short time relative to the total expected duration of the exercise; in this case, the value will not usually vary enough between the beginning of an action and the end to make this overly inaccurate. Under these assumptions the total value function is still separable, but the assumptions are rather stronger.

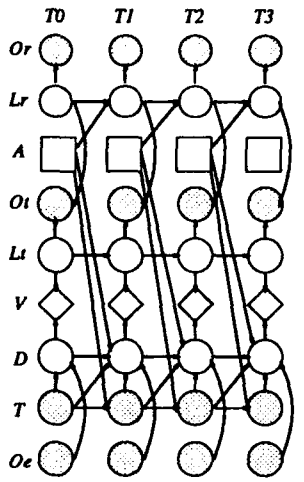


Figure 2: FTR model with variable time intervals

Consider an alternative specification of the problem: again, the robot is required to locate the missile silo, but now it must choose when to report and it is allowed to report exactly once. Moreover, there is an additional consequence of being detected: the robot cannot report on the location of the target if it has been detected, adding an incentive not to delay in making a report. As far as we can tell, any reasonable value model for this problem will not admit a time-separable value function.

Methods for Reducing Complexity

Bayesian networks provide a concise and easily-formulated representation of state and causality in many domains. However, the complexity of evaluation of these networks makes it unrealistic to use them unless the size of the network and number of states for each node are fairly small. Some information about the world cannot be easily represented in this form without a large number of nodes and/or large state spaces, yet an approximation of this information can often be used by dynamically modifying the conditional probabilities represented in a network.

This kind of situation arises in the FTR problem described above. Consider the case where the robot must explore a room, looking for a particular kind of object.

We assume that one of the sensors available to the robot allows it to gather information about the presence of the object; for example, a vision system might be used to provide a degree of belief about the target being present in a particular image. We assume also that the robot has collision sensors, for example sonar transducers that allow it to form a coarse map of the room in order to move around without colliding with furniture.

If the layout of the room is known in advance, along with the location of the furniture and anything else in the room, then we can construct a network that allows us to represent the location of the robot, that of the target, and the observations about the target. Given enough knowledge about the contents of the room, we can estimate the conditional probabilities for the nodes in this network.

More realistically, we will have little or no advance knowledge of the contents of the room, and quantifying the network will be impossible to do accurately. In particular, given a location for the robot, and a location for the target, we need to specify the probabilities of the different possible observations the robot will make of the target. Clearly this depends heavily on what is between the target and the robot.

However, once the robot is in the room making the observations that it needs to make in order to move about, extra evidence is accumulating that could be used to refine these conditional probabilities. There are two ways to exploit this accumulating information. One is to explicitly represent our knowledge of the contents of the room in the network. As we shall see, this is likely to be prohibitively expensive from a computational standpoint. The other is to specify the conditional probabilities in the network as a function of our knowledge about the room, and to update these conditional probabilities as we gather more information about the room. This does not allow us to exploit the information as fully as if it were explicitly quantified, but it provides us with a good approximation at very little computational cost.

A common technique for gathering evidence about obstacles in an open area such as a room is to use a volumetric spatial representation such as an occupancy grid [Moravec and Elfes, 1985]. The area is divided into (usually equally sized) grid elements, and a probability that each element contains an obstacle is maintained. As evidence from sensors is gathered, it is used to update these probabilities.

It is possible to represent the same information using a Bayesian network. The resulting network, for a very coarse grid, might look like figure 3. We have restricted ourselves to a single time-slice, as otherwise the network is too highly connected to be displayed in any reasonable way. Another external observation node has been added: *OccObs*. It represents the observations made about occupancy, for example from sonar readings made by the robot. For presentation purposes,

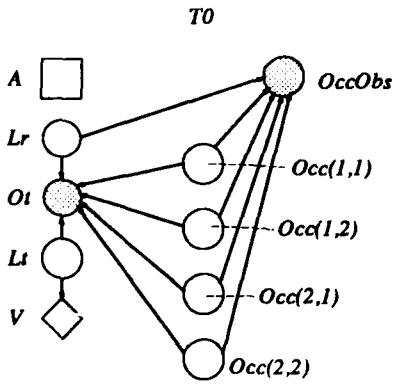


Figure 3: Searching a room using an occupancy grid

we have represented the case where the grid is 2 by 2, which would in reality be much too coarse to obtain any useful information. Four state nodes have been added, one for each grid cell. Each of these nodes represents whether there is an object in that grid element that would occlude the target were it between the robot and the target.

We use the term occlusion in the broad sense of inhibiting observation; here, we assume that the robot is using some form of vision to detect the object, so we are interested in whether grid elements contain an opaque object that will make it less likely that the robot will see the target if it is behind it. The same basic information can be used by the robot to determine places to hide in order to avoid detection. Occupancy information can be with respect to a coordinate system centered on the robot or with respect to some other, perhaps global coordinate system.

Now at each time-step, observations are made about the target, as before, but also about the presence of objects in the room that may occlude the target. The latter observations will affect the probability distribution on the node O_t since they will influence the distributions of the *Occ* nodes, which influence O_t. Moreover, the system can reason about change in the distributions of these *Occ* nodes.

This is a very appealing model — it allows us to reason about change in the robot's knowledge about occupied regions of space; we can represent the likelihood that obstacles will block the target from view, and we are using information that we would have to gather anyway in order to move around the room. Unfortunately, adding one node per grid element per time slice makes the network enormously expensive to evaluate and update in response to new evidence. Given the computational complexity inherent in representing this information explicitly, we consider the possibility of using the information in a less completely integrated manner, at a much smaller computational cost.

For a given set of probabilities in the occupancy grid,

specifying the conditional probabilities in the network is straightforward — this is effectively the situation when the room's contents are known in advance. We propose that a network such as the one in Figure 1 be used to model the room the robot is searching. Simultaneously, an occupancy grid is constructed as the robot explores the room, independently of the Bayesian network. At each time step, before projecting the results of action sequences to choose the next action to perform, we update the conditional probabilities in the network in light of the information in the occupancy grid. The portions of the network that will change given new information about occupancy are: the distribution of O_t given L_t and L_r, and the priors for L_r and L_t at the first time-slice in the network (i.e. now). More specifically, the distribution of O_t given L_t and L_r will depend on the values in the occupancy grid between L_t and L_r — the more likely there is to be an obstacle the less likely the robot is to see the target. The priors for L_r and L_t may be influenced by the contents of the occupancy grid if for example we know in advance that the robot is unlikely to be in the same location as an obstacle, or that the target is likely to be on an obstacle such as a desk.

This makes the specification of the probability distributions more complex for the designer of the network, since they will be parameterized by the values in the occupancy grid, but the amount of information available with which to make inferences is much larger at run-time.

One disadvantage of this dynamic technique over explicitly representing the grid in the network is that the system cannot reason about the effect of its actions on the contents of the grid, as described in the section above. However, given that representing this occupancy information makes the decision model computationally intractable in all but trivial cases, the choice is between ignoring the effect of occupancy on the observations made by the robot and integrating them by re-quantifying the network; clearly the latter provides us with a better model.

Related Work

There is a growing literature on using probabilistic networks for applications in planning, interpretation, and image understanding. Levitt *et al.* [1988] discuss search issues in the context of object recognition. Agosta [1991] discusses the issues involved in quantifying relations among visual features. Goldman [1990] addresses related problems in the context of natural language understanding, with an emphasis on dynamically generating probabilistic models.

Conclusions

This paper presents techniques for applying a well known, mathematically pleasing but computationally impractical approach for decision making under uncertainty to problems in robotics and perception. We provide practical representational alternatives for reasoning about the fundamental issues of time, space, and decision value. Ultimately, we are looking for a comprehensive and disciplined approach to making the sort of tradeoffs involving modeling precision and computational complexity that are outlined in this paper. While that goal is still some distance off, we are beginning to understand some of the ramifications of various design decisions.

References

[Agosta, 1991] Agosta, J. M. 1991. Conditional inter-causally independent' node distributions, a property of 'noisy-OR' models. In *Proceedings of the Seventh Conference on Uncertainty in AI.* 9–16.

[Dean and Kanazawa, 1989] Dean, Thomas and Kanazawa, Keiji 1989. A model for reasoning about persistence and causation. *Computational Intelligence* 5(3):142–150.

[Dean and Wellman, 1991] Dean, Thomas and Wellman, Michael 1991. *Planning and Control.* Morgan Kaufmann, San Mateo, California.

[Durrant-Whyte, 1988] Durrant-Whyte, Hugh F. 1988. *Integration, Coordination and Control of Multi-Sensor Robot Systems.* Kluwer Academic Publishers, Boston, Massachusetts.

[Goldman, 1990] Goldman, Robert 1990. *A Probabilistic Approach to Language Understanding.* Ph.D. Dissertation, Brown University, Providence, RI. Available as Technical Report CS-TR-90-34.

[Hager, 1990] Hager, Gregory D. 1990. *Task-Directed Sensor Fusion and Planning: A Computational Approach.* Kluwer Academic Publishers, Boston, Massachusetts.

[Howard and Matheson, 1984] Howard, Ronald A. and Matheson, James E. 1984. Influence diagrams. In Howard, Ronald A. and Matheson, James E., editors 1984, *The Principles and Applications of Decision Analysis.* Strategic Decisions Group, Menlo Park, CA 94025.

[Levitt *et al.*, 1988] Levitt, Tod; Binford, Thomas; Ettinger, Gil; and Gelband, Patrice 1988. Utility-based control for computer vision. In *Proceedings of the 1988 Workshop on Uncertainty in Artificial Intelligence.*

[Moravec and Elfes, 1985] Moravec, H. P. and Elfes, A. 1985. High resolution maps from wide angle sonar. In *IEEE International Conference on Robotics and Automation.* 138–145.

[Pearl, 1988] Pearl, Judea 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan-Kaufmann, Los Altos, California.

[Peot and Shachter, 1991] Peot, Mark and Shachter, Ross 1991. Fusion and propagation with multiple observations in belief networks. *Artificial Intelligence* 48(3):299–318.

Bounded and Unbounded Implicit Polynomial Curves and Surfaces, Mahalanobis Distances, and Geometric Invariants, for Robust Object Recognition *

Daniel Keren, Jayashree Subrahmonia, Gabriel Taubin+, David B. Cooper
Laboratory for Engineering Man/Machine Systems
Division of Engineering, Brown University, Providence, RI 02912

+IBM T.J.Watson Research Center, P.O.Box 704 (H1-E48), Yorktown Heights, NY 10598

Abstract

Implicit higher degree polynomials in x, y, z (or in x, y for curves in images) have considerable global and semiglobal representation power for objects in 3D space. (Spheres, cylinders, cones and planes are special cases of such polynomials restricted to second degree.) Hence, they have great potential for object recognition and position estimation. In this paper we deal with three problems pertinent to using these polynomials in real world robust systems: 1) Characterization and fitting algorithms for the subset of these algebraic curves and surfaces that is bounded and exists largely in the vicinity of the data; 2) A Mahalanobis distance for comparing the coefficients of two polynomials to determine whether the curves or surfaces that they represent are close over a specified region; 3) Geometric Invariants for determining whether one implicit polynomial curve or surface is a rotation and translation of another, or whether one implicit polynomial curve is an affine transformation of another. In addition to handling objects with easily detectable features such as vertices, high curvature points, and straight lines, the polynomials and tools discussed in this paper are ideally suited to smooth curves and smooth curved surfaces which do not have detectable features.

1 Introduction and Previous Work

Much of the early work on implicit polynomial curves and surfaces were limited to 2nd degree polynomials, thus dealing with representations that had modest expressive power, but the fitting algorithms were simple, the computational cost small, and the resulting polynomial coefficients were reasonably stable [4, 17, 16, 13, 5, 3, 14]. Implicit polynomial curves and surfaces of degree higher than two have great modeling power for complicated shapes and can be made to fit data very well, but their coefficients may be sensitive to small changes in the data. This poses a problem since we would like to compare curves and surfaces based on their polynomial coefficients or functions of the coefficents that represent only shape, i.e. that are invariant to object rotation, translation and stretching in two directions – general linear transformations. In this paper we present new approaches and tools to these problems which should permit robust 2D curve and 3D surface object recognition and position estimation based on the *polynomial coefficients only*. In particular, we introduce the class of implicit polynomials that represent closed curves or surfaces, exhibit a low computation cost algorithm for fitting such that the curve or surface exsists only in the general region of the data, illustrate the wide range of shapes that can be represented and illustrate the improved stability of the coefficients. Then, for any polynomial whether it represents a closed or an open unbounded curve or surface, we present and discuss a simple expression for the aposteriori probability distribution of its coefficients given the data set that is to be represented by a polynomial. Polynomial coefficient sensitivity to small changes in the data occurs when a data set does not sufficiently constrain the coefficients. Our development both determines the subset of coefficient space that is constrained by the data and provides the appropriate metric for polynomial curve or surface recognition based on the polynomial coefficient vectors. Lastly, two approaches are discussed to the design of functions of the polynomial coefficients that are invariant to Euclidean or affine transformations. Both approaches appear to be promising and can be functions of all the coefficients rather than of just the coefficients of higher degree, which is the subject of classical geometric invariance theory.

2 Description of Closed Objects Using Polynomials

2.1 Finding the Fitting Polynomial

In general, for a polynomial $p(x, y)$ to describe a closed object O with boundary B the following should hold:

1) The set $\{(x, y) : p(x, y) = 0\}$ is equal to B.

2) $(x, y) \in O$ iff $p(x, y) < 0$.

We shall refer to the set $\{(x, y) : p(x, y) = 0\}$ as the *zero set*. Let us note at this point that polynomials with an unbounded zero set can describe curve patches, but in Section 2 we are interested in describing closed bounded objects.

*This work was partially supported by NSF Grant #IRI-8715774 and NSF-DARPA Grant #IRI-8905436

Since second degree polynomials can describe only circles and ellipses, let us proceed to higher degrees. The standard notation for a polynomial of degree n will be adopted: $p(x,y) = \sum_{0 \le i+j \le n} a_{ij} x^i y^j$. The following simple lemma shows that the next class in the polynomial hierarchy is not suitable for describing closed objects.

Lemma 1 *The zero set of a third degree polynomial is unbounded.*

Proof: [11].

Next on the list are fourth degree polynomials. Their zero set can be bounded – e.g. $x^4 + y^4 - 1 = 0$ – or unbounded, e.g. $x^4 - y^4 = 0$. It is not surprising that the high powers of the polynomial determine if its zero set is bounded or not. Let us call those fourth degree powers, e.g, $a_{40}x^4 + a_{31}x^3y + a_{22}x^2y^2 + a_{13}xy^3 + a_{04}y^4$, the *leading form* of $p(x,y)$, or $p_4(x,y)$, and the sum of the lower powers – e.g. cubics.quadrics,linear terms and the constant – the *lower terms* or $p_3(x,y)$. Let us also define a polynomial to be *stably bounded* if a small perturbation of its coefficients leaves its zero set bounded. For reasons of numerical robustness we are interested only in stably bounded polynomials.

Theorem 1 *The zero set of $p(x,y)$ is stably bounded iff there exists a symmetric positive definite matrix A such that $p_4(x,y) = (x^2\ xy\ y^2)A(x^2\ xy\ y^2)^T$.*

Proof: [11]. Summerizing, given an object O with boundary B we look for a fourth degree polynomial $p(x,y)$ such that:

1) $p_4(x,y)$ can be expressed as $(x^2\ xy\ y^2)A(x^2\ xy\ y^2)^T$ with A symmetric positive definite.

2) The zero set of $p(x,y)$ approximates B.

We know what the first condition means. How is the second satisfied? The first guess is: find $p(x,y)$ such that $\sum_{(x_i,y_i)\in B} p^2(x_i,y_i)$ is minimal. This, however, results in far from optimal description of B because $p^2(x_i,y_i)$ is often a poor measure for the distance of (x_i,y_i) from the zero set of $p(x,y)$. A much better measure, suggested by [17] and extended in [21], is $\frac{p^2(x_i,y_i)}{\nabla\ p(x_i,y_i)}$ the approximate squared distance of (x_i,y_i) from the curve, (where $\nabla^2 p(x_i,y_i)$ stands for the square of the norm of the gradient). So the expression to be minimized is

$$\sum_{(x_i,y_i)\in B} \frac{p^2(x_i,y_i)}{\nabla^2 p(x_i,y_i)} \qquad (1)$$

Taubin [21], in an extensive work on implicit curve and surface fitting, solves the problem by approximating Equation 1 with the expression

$$\frac{\sum_{(x_i,y_i)\in B} p^2(x_i,y_i)}{\sum_{(x_i,y_i)\in B} \nabla^2 p(x_i,y_i)} \qquad (2)$$

and then minimizing Equation 2 by generalized eigen vector techniques, followed by an iterative scheme for improving the polynomial fit. Taubin's work results in excellent fits, but he does not worry about the zero set being bounded; hence the outcome of his fitting algorithm is that the zero set *contains* B but often has additional unbounded parts (see Figures 1,2). A simple example is that of a square; Taubin's algorithm describes it as the union of four straight lines, with the corresponding $p(x,y)$ equal to the product of the four linear polynomials describing these lines. So the square is represented as the union of the infinite extension of its edges.

The question is how to incorporate into Taubin's algorithm the condition that the zero set be bounded. What should be done is simple: look only for polynomials $p(x,y)$ such that $p_4(x,y)$ can be expressed as in Theorem 1. The question is how to parametrize positive definite matrices. We use the following result [1]: if a matrix A is symmetric positive definite, it has a symmetric *square root* B.

Hence it is enough to look at all $p(x,y)$'s where $p_4(x,y)$ can be written as $(x^2\ xy\ y^2)B^2(x^2\ xy\ y^2)^T$ where B is symmetric. Thus the strategy chosen was to minimize the error measure of Equation 1 while conforming to the above condition. This is done by minimizing not over the space of unconstrained polynomials, but only over the space of $p(x,y)$ such that $p_3(x,y)$ is unconstrained and $p_4(x,y)$ is as above. Technically, we look for the optimal B (6 parameters) and $p_3(x,y)$ (10 parameters). Note that in Equation 1, fourth powers of the elements of B appear. This does make the minimization problem non-linear, but that seems a reasonable price to pay if one wants to enforce boundness of the zero set. Sometimes, the zero set is bounded but contains spurious parts (Figure 5b). This problem was solved by adding ϵI to B^2, where I is the identity matrix and ϵ a small positive constant.

Another problem affecting the running time of the fitting algorithm is that the expression in Equation 1 is expensive to calculate. Most non-linear minimization techniques require computation of the function and its derivatives many times. If many points are present, this means computing the sum of the function over its gradient squared in all these points, requiring enormous time. However, it is possible to overcome this problem using the following iterative algorithm:

1) minimize $\sum_{(x_i,y_i)\in B} p^2(x_i,y_i)$. This is quite fast, because the sum of the squares of the polynomial at the points can be written as FMF^T where F is the vector of the polynomial's coefficients and M is a scatter matrix of the points [21]. This is much faster than using Equation 1 directly. Call the optimal polynomial $P_1(x,y)$.

2) Assign to each point p_i a weight $w_i = \frac{1}{\nabla^2 P_1(p_i)}$.

3) Minimize $\sum w_i P^2(p_i)$. This is also quick – it is exactly the same process as in **1)**, with M replaced by a weighted scatter matrix.

4)Go back to **2)** and update the weights using the minimizer of **3)** instead of $P_1(x,y)$.

5) Iterate untill the error of fit, measured by Equation 1, doesn't decrease substantialy. (Note that we are using Equation 1, but only a small number of times – usually less than 5 iteration are needed).

This algorithm is suboptimal in the sense that it doesn't minimize Equation 1, but it is much faster and results in the same quality of fits. Running times will be discussed in Section 3.

Some examples are provided of bounded vs. unbounded descriptions. In Figures 1 and 2 an object (super-quadric) is shown with bounded and unbounded polynomial fits (the scale has been changed to show the global behavior of the zero set of the unbounded polynomial). It is interesting to observe how the zero set in Figure 2 contains the super-quadric but has additional unbounded components. In Figure 3 an assortment of objects that can be exactly describe by fourth degree polynomials is presented, and in Figure 4 the power of polynomials is demonstrated by showing four disconnected objects that are represented by a single fourth degree polynomial.

Only polynomials of degree four have been used in this work. In the future, higher order polynomials will be investigated.

2.1.1 Description of 3D Objects

Everything that was said about polynomials $p(x, y)$ extends to $p(x, y, z)$. Boundenss of the zero set can be satisfied by using the identity for a fourth-degree polynomial:

$$(x^2\ xy\ xz\ y^2\ yz\ z^2)A(x^2\ xy\ xz\ y^2\ yz\ z^2)^T$$

$$= a_{400}x^4 + a_{310}x^3y + a_{301}x^3z + ...$$

where A is a 6×6 symmetric positive definite matrix, and proceeding with the same line of thought as for $p(x, y)$. In Section 3 results for 3D fits will be surveyed.

2.2 Relation to Other Models: Super-Quadrics

An analytic model that has proved effective in describing non-polygonal objects in Graphics, Vision, and Robotics is the *super-quadric* model [2, 9]. As the name implies these are extensions of ellipses and ellipsoids (and indeed are also called super-ellipsoids). In the plane, all super-quadrics can be described by a scaling, rotation, and translation of the "twisted circles" $C_\epsilon = \{(x, y) : x^\epsilon + y^\epsilon = 1\}$. If $\epsilon = 2$, C_ϵ is a circle; for $\epsilon > 2$ it is a circle that is "pushed out", approaching the unit square as ϵ grows; for small ϵ, C_ϵ is a non-convex star-shaped set (see Figure 1).

It can be seen that an unconstrained fourth degree fit to the super-quadric in Figure 2 is indistinguishable from the super-quadric, but its zero set is unbounded. Depending on the application, this may or may not pose a problem. How do fourth degree polynomials with a closed zero-set relate to super-quadrics? Since polynomials can also be scaled, rotated, and translated, then to relate the description power of polynomials to that of super-quadrics it is enough to check if the C_ϵ are well approximated by polynomials. The approximation for $\epsilon = 0.6$ is presented in Figure 1. For larger values of ϵ the approximations are so good that one cannot distinguish the super-quadric from the fitting polynomial by looking at them! It can be seen that the polynomials give a reasonably good description of the super-quadrics for small ϵ, although they miss the sharp protrusion coming out of the sides of the super-quadrics. These results should be expected since the polynomials have more degrees of freedom than the super-quadrics.

3 Invariants: Using Polynomials for Recognition

3.1 What are Invariants?

Invariants are quantities assigned to polynomials that do not change when the coordinate system undergoes transformations, and hence are descriptors of shape a natural candidate for recognition purposes. Probably the first invariants known were those of the second degree polynomial $a_{20}x^2 + a_{11}xy + a_{02}y^2 + a_{10}x + a_{01}y + a_{00}$: if the coordinate system undergoes an affine transformation $(u, v)^t = A(x, y)^t + T$, where A is a non-sigular matrix and T a vector, then the quantities $a_{20} + a_{02}$ and $4a_{20}a_{02} - a_{11}^2$ are multiplied by the square of the determinant of A. Classical work [8] was concentrated on finding affine invariants, but only of the leading form. This is probably because the leading form does not change under translation, hence the problem of finding invariants is more approachable and the theory more elegant. In [8], a beautiful solution - the *symbolic method* - is presented for writing down all the affine invariants of the leading form. However, there are other classes of invariants, such as Euclidean invariants, and invariants that depend on *all* the coefficients, that remain to be explored. Also, as noted in [19, 20, 21], in real applications one should use invariants that are as simple as possible. Taubin finds many invariants that are easy to compute, as they are expressed as eigenvectors of relatively small matrices. However, for application of Section 4 to invariants, they pose a problem, since they are not given as *explicit* functions of the coefficients. and thus cannot be used directly in the statistical analysis presented there. This is because the goal is to utilize probability distributions on the coefficients to find the probability distribution of the invariants; for that, simple explicit invariants are necessary. [6, 7] contain an extensive treatment of explicit invariants for Vision for second degree polynomials curves.

3.2 Using Symbolic Computation to Find Invariants

In order to find simple and explicit invariants of polynomials, the tool of symbolic computation was used. We have used the Mathematica package [22], running on a Sparc working station. Recently, symbolic computation is finding applications to Vision [12, 10].

The method used to find invariants is best demonstrated by a simple example. The input for Mathematica to find Euclidean invariants of the second degree form in two variables x and y is presented next:

1 $a_{20}x^2 + a_{11}xy + a_{02}y^2$
2 %/.$x- > u(1 - q^2/2) - v(q - q^3/6)$
3 %/.$y- > v(1 - q^2/2) + u(q - q^3/6)$
4 expl=Expand[%]
5 aa_{20} = $D[expl, \{u, 2\}]/2$
6 aa_{02} = $D[expl, \{v, 2\}]/2$

7 $aa_{11} = D[exp1, \{u, 1\}, \{v, 1\}]$
8 $exp2 = Expand[Aaa_{20}^2 + Baa_{11}^2 + Caa_{02}^2 + Daa_{20}aa_{11} + Eaa_{20}aa_{02} + Faa_{11}aa_{02}]$
9 $exp3 = Aa_{20}^2 + Ba_{11}^2 + Ca_{02}^2 + Da_{20}a_{11} + Ea_{20}a_{02} + Fa_{11}a_{02}$
10 exp4=Expand[exp2-exp3]
11 $q1 = Expand[D[exp4, \{a_{20}, 2\}]]$
12 $q2 = Expand[D[exp4, \{a_{11}, 2\}]]$
13 $q3 = Expand[D[exp4, \{a_{11}, 2\}]]$
14 $q4 = Expand[D[exp4, \{a_{20}, 1\}, \{a_{11}, 1\}]]$
15 $q5 = Expand[D[exp4, \{a_{20}, 1\}, \{a_{02}, 1\}]]$
16 $q6 = Expand[D[exp4, \{a_{11}, 1\}, \{a_{02}, 1\}]]$
17 eq1=(Coefficient[q1,q,0]==0)
18 eq2=(Coefficient[q1,q,1]==0)
.
.
.
19 eq3=(Coefficient[q1,q,2]==0)
20 eq22=(Coefficient[q6,q,1]==0)
21 eq23=(Coefficient[q6,q,2]==0)
22 eq24=(Coefficient[q6,q,3]==0)
23 Solve[{eq1,eq2,...,eq23,eq24},{A,B,C,D,E,F}]

In line 1, a leading form of degree 2 in two variables is presented. In lines 2,3,4 the coordinates are rotated using Taylor series approximati ons for the trigonometric functions. In lines 5,6,7 the coefficients of the new form are computed. in Lines 9,10 we *guess* for invariants that are second degree polynomials in the coefficients; there is no formal justific ation for that, but there seem to be many invariants that are of this type. Let us call the power of the coefficients in the invariant the *rank* of the invariant; thus the invariant $a_{20} + a_{02}$ is of rank one, and $4a_{20}a_{02} - a_{11}^2$ of rank two. In lines 10-16, the difference between the polynomial expression in the new coefficients (exp2) and the old ones (exp3) is differentiated so as to get the coefficients out of the way, and in lines 17-22 the derivatives of these expressions by q are equaled to zero (it is necessary to get rid of all variables in order to remain with a *linear* system of equations). In line 23 the system is solved. After 4.5 seconds, Mathematica gave the following answer:

$$\{A = C,\ B = C/2 - E/4,\ D = 0,\ F = 0\}$$

Substituting first $\{C = 1,\ E = 0\}$ and then $\{C = 0,\ E = 1\}$ we get two invariants that are equivalent to the invariants noted before - $(a_{20} + a_{02})^2$ and $4a_{20}a_{02} - a_{11}^2$ (remembering that combinations of invariants are also invariants).

We note here that solutions obtained this way are not necessarily invariants, because the trigonometric functions were replaced by approximations. However, the method does assure that all invariants of the particular shape sought are included in the set of solutions, and what is left is to detect if any spurious solutions crept in. Fortunately, it is much easier to check if a certain expression is an invariant then to find invariants; to do that, change the coordinate system and check if the expression remains the same. In all our experiments we did not come on any spurious solutions. An intuitive explanation is that expressions that are invariant under arbitrarily small rotations are invariant under every rotation, as every rotation can be generated by composing small rotations. For affine invariants we have to force the expression to stay constant under translation also, but the idea is the same.

The example above is trivial. However, this simple technique gets rather messy when searching for invariants of higher rank of polynomials of say fourth degree, and invariants of polynomials in 3 variables. In general, a d-th degree form in n variables has $\binom{n+d-1}{d}$ coefficients. Suppose we want to find out if the third degree form in x, y, z, (e.g. $a_{300}x^3 + a_{210}x^2y + .. + a_{003}z^3$ (10 coefficients) has invariants of rank four. Then the number of the coefficients which are the analogues of A, B, C, D, E, F in the example above is $\binom{10+4-1}{4} = 715$. It is easy to see that manually writing down the system of equations becomes impossible, and indeed it was done using code generating programs. The resulting system of equations contained 4,290 linear equations, and the time of handling the huge expressions and solving the system was about 5 hours. Many more invariants were found this way, including invariants that are not of the leading form but of all the coefficients (there is no known method for finding these) of polynomials in two and three variables. If *affine* invariants are sought, the calculation can be made much easier by noting that the possible invariants are only a small subset of all polynomials in the coefficients; looking for instance at the second degree form in x, y, we can rule out immediately an invariant containing both a_{20}^2 and $a_{11}a_{02}$. This is because stretching the coordinate system by c in the x-direction would multiply a_{20}^2 by c^4, and $a_{11}a_{02}$ by c. So, the only combinations that can constitute an invariant are $a_{20}a_{02}$ and a_{11}^2. Similar considerations apply for other polynomials. For instance, finding affine invariants of rank four of a fourth-degree polynomial in x, y was only a few minutes work for Mathematica. A simpler example, finding Euclidean invariants of rank two of a fourth-degree polynomial in x, y , took 20 seconds and resulted in the invariants

$$a_{22}^2 - 3a_{13}a_{31} + 12a_{04}a_{40},$$

$$3a_{13}^2 - 8a_{04}a_{22} + 2a_{13}a_{31} + 3a_{31}^2 - 32a_{40}a_{04} - 8a_{22}a_{40},$$

$$3a_{04}^2 + 2a_{04}a_{22} + a_{13}a_{31} + 2a_{04}a_{40} + 2a_{22}a_{40} + 3a_{40}^2,$$

Of these, only the first one is an affine invariant and hence is the only one that can be found using the symbolic method. In addition, 8 invariants of rank 3 of a fourth-degree polynomial in x, y were found.

Summerizing, the brute force technique of using symbolic computations by computer for transforming the invariant problem to a (huge) linear system proved very useful in finding simple, explicit invariants that cannot be found by the symbolic method, such as Euclidean invariants and invariants that depend on all coefficients and not only on the leading form.

An example is given for recognizing an object with polynomials. Note the object does not have any of the common features often used such as vertices or high curvature points and recognition methods based on feature identification might run into trouble trying to recognize

objects of this type. Figure 5a shows the data set, the bounded 4th degree polynomial fit and three Euclidean invariants discussed in section 3. Figure 5b shows the data set rotated and translated and a 4th degree polynomial fit that is not restricted to lie in the vicinity of the data. Although the polynomial in this figure fits the data very well, the invariants change somewhat. In Figure 5c, the same data set as in 5b is fit with a 4th degree polynomial while restricting the zero set of the polynomial to lie within the vicinity of the data [11]. Note that the invariants are closer to the invariants of figure 5a. In Figure 5d, the data set is further rotated and contaminated with considerable noise. Again the zero set is restricted to lie in the data region, and the resulting invariants are quite similar to those in 5a. These sets of figures illustrate that first, the invariants discussed in this section really work, and second, restricting the polynomials to the vicinity of the data set can help to stabilize the estimated coefficients, and, hence, the invariants. Figure 5e shows a different object with its fourth degree fit and the invariants, and Figure 5f a rotation and translation of that data with its invariants (please note that the data sets in Figures 5a-d and 5e-f are the same, and look different because of scaling by the drawing package used). In practice we envision using a larger number of invariants.

3.3 Experiments

Experiments were run on two and three dimensional data to test the fitting algorithm. The minimization scheme to solve the non-linear optimization problem was Powell's method [15]. Implementation was quite simple, and the programs for fitting 2D and 3D data consisted of about 500 and 800 lines in C respectively, which were compiled and executed on a SPARC 2. A nice feature of the algorithm is that no initial guess is needed – all iterations started with the zero polynomial.

The first example consists of 175 points in the shape of a packman. The first three iterations are shown in Figure 6 super-imposed on the data. After the third iteration, there is no significant improvement in the error. The error is larger than in the other examples, but that is because of the very rough nature of the data points.

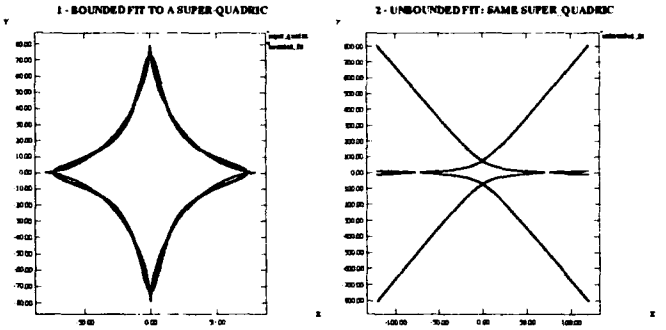
The second example consists of 442 points that lie on a sphere of radius 50 around the origin and 442 points on a sphere of radius 100 around the origin. Since the union of the spheres can be exactly described by an implicit polynomial of degree four, the resulting error is very small.

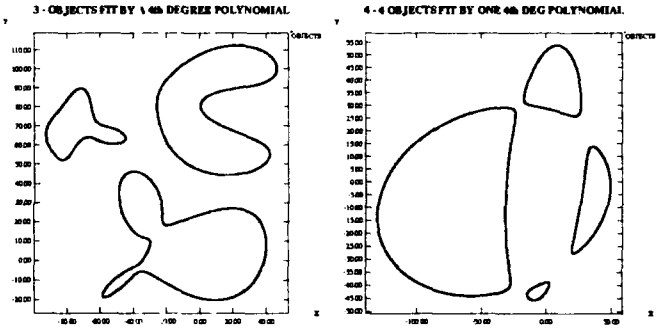
The third example (super quadric) consists of 441 points on the surface of a super quadric parametrized as following –

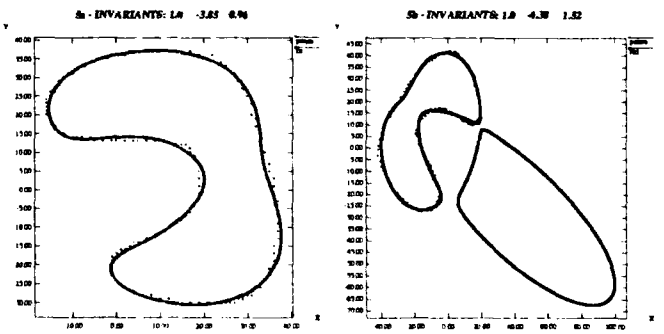
$x = 50\cos^{0.4}(\phi)\cos^{0.6}(\theta)$, $y = 70\cos^{0.4}(\phi)\sin^{0.6}(\theta)$, $z = 100\sin^{0.4}(\phi)$

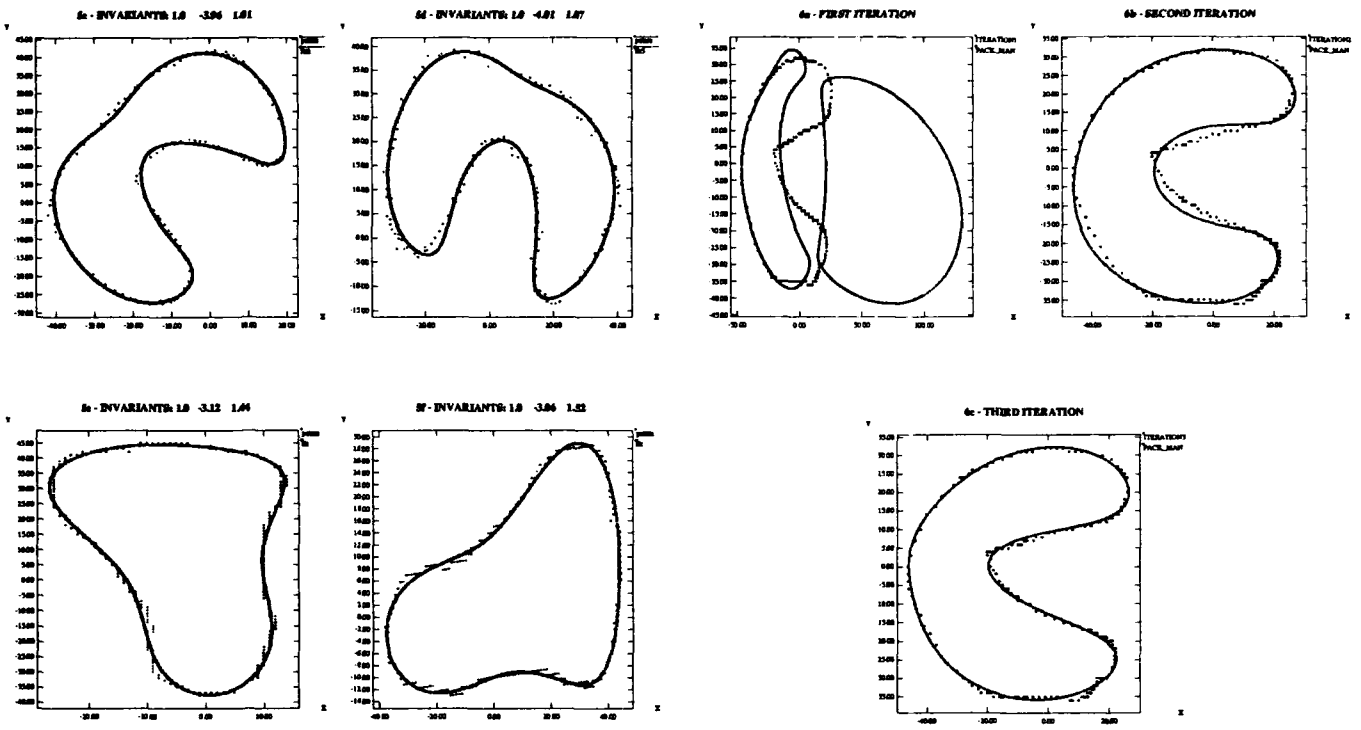
Results are given in the table with the 1,2,3 on the left corresponding to the number of the iteration. Time is in seconds. The error is the average distance of the data points from the fitted surface. Compared to the size of the objects, the errors are small.

	Packman (175 points)		Spheres (884 points)		Super quadric (441 points)	
	time	error	time	error	time	error
1	0.9	4.9	17.3	1.3	10.3	0.14
2	1.1	1.4	43.2	$3 \cdot 10^{-6}$	12.7	0.08
3	1.4	0.68	-	-	-	-









3.4 Eigenvalues as Invariants

In [19, 21] many new computationaly efficient invariants are presented. We briefly give some of the underlying ideas behind the theory. The following is well known.

The quadratic form, i.e., 2nd degree terms, in two variables can be expressed as

$$\phi(x_1, x_2) = \frac{1}{2}(x_1, x_2) \begin{pmatrix} \Phi_{(2,0)} & \Phi_{(1,1)} \\ \Phi_{(1,1)} & \Phi_{(0,2)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

and denoting the 2×2 matrix as $\Phi_{[1,1]}$, we can write $\phi(x) = \frac{1}{2}x^t \Phi_{[1,1]} x$. Now if $x' = Ax$ is a linear transformation, the transformed polynomial, $\phi'(x')$, satisfies

$$\phi'(x') = \phi(A^{-1}x') = \frac{1}{2}x'^t \Phi'_{[1,1]} x' ;$$

hence, $\Phi'_{[1,1]} = A^{-t}\Phi_{[1,1]}A^{-1}$, which reduces to $\Phi'_{[1,1]} = A\Phi_{[1,1]}A^t$ if A is orthogonal, i.e., a rotation. Since the determinant $|A|$ is 1 for A orthogonal, we see that $|\Phi_{[1,1]}|$ is invariant to object rotation. Furthermore, it is easily seen that the coefficients of the characteristic polynomial $\chi(\lambda) = \det(\lambda I - \Phi_{[1,1]})$ are orthogonal invariants of the form ϕ. Equivalently, the two eigenvalues of the matrix, the roots of $\chi(\lambda)$, are orthogonal invariants. For A nonsingular we have $|\Phi_{[1,1]}| = |A|^{-2}|\Phi'_{[1,1]}|$. Let $\phi(x) = \frac{1}{2}x^t\Phi_{[1,1]}x$ and $\psi(x) = \frac{1}{2}x^t\Psi_{[1,1]}x$ be two nonsingular quadratic forms in three variables, x, y, z. It is well known that $\text{trace}(\Phi_{[1,1]}\Psi^{-1}_{[1,1]})$ and $\det(\Phi_{[1,1]}\Psi^{-1}_{[1,1]})$ are joint rational (ratio of two polynomials) linear invariants of the two forms. In fact, these two invariants are just two of the coefficients of the characteristic polynomial $\chi(\lambda) = \det(\lambda I - \Phi_{[1,1]}\Psi^{-1}_{[1,1]})$, and all the coefficients of $\chi(\lambda)$ are joint linear invariants of the two forms. Equivalently, the three eigenvalues of the matrix $\Phi_{[1,1]}\Psi^{-1}_{[1,1]}$ are joint invariants of the two forms. That is, the eigenvalues are not only invariant to rotations of the 3D or 2D object, but also to arbitrary stretchings in two arbitrary directions. Finally, this construction can be extended to all the pairs of quadratic forms, not only the nonsingular ones: instead of considering the eigenvalues of $\Phi_{[1,1]}\Psi^{-1}_{[1,1]}$, consider the generalized eigenvalues of the pair of matrices, the values of $(\lambda, \nu) \neq 0$ such that $\det(\lambda\Phi_{[1,1]} + \nu\Psi_{[1,1]}) = 0$.

In [19, 21] Taubin generalizes the preceding constructions to higher degrees. That is, he shows how to compute orthogonal and linear invariants of one form or of two or more forms of different degrees, jointly by reducing the problem to the computation of eigenvalues or generalized eigenvalues of certain matrices of coefficients. Thus, the invariants can be designed to be functions of the coefficients of all the terms in a polynomial or of just those terms of one or a few degrees. Taubin's development exploits the following known results.

The set of monomials $\{ x^\alpha/\sqrt{\alpha!} : |\alpha| = d \}$ of degree d lexicographically ordered, define a vector of dimension h_d, which we will denote $X_{[d]}(x)$. For example,

$$X_{[3]}(x_1, x_2) = \left(\tfrac{1}{\sqrt{6}}x_1^3 \quad \tfrac{1}{\sqrt{2}}x_1^2x_2 \quad \tfrac{1}{\sqrt{2}}x_1x_2^2 \quad \tfrac{1}{\sqrt{6}}x_2^3 \right)^t .$$

Hence, $X_{[3]}(x_1, x_2)$ is the vector of monomials of degree 3 for points x_1, x_2 in the plane. Next, any form of degree d=j+k, i.e., polynomial where all the terms have degree d resulting from the products of monomials of degrees j and k, can be expressed as

$$\binom{d}{j}\phi(x) = X_{[j]}(x)^t \Phi_{[j,k]} X_{[k]}(x) .$$

Finally, if $x' = Ax$ *is a nonsingular linear transformation*, the resulting polynomial is $X_{[j]}(x')^t \Phi'_{[j,k]} X_{[k]}(x')$, with

$$\Phi'_{[j,k]} = A^{-t}_{[j]} \Phi_{[j,k]} A^{-1}_{[k]} .$$

where the elements of $A_{[j]}$ and $A_{[k]}$ depend only on A. Using this result, Taubin develops a rich range of easily computable invariants for polynomial curves and surfaces of arbitrary degrees.

4 Asymptotic Parameter Distributions, Mahalanobis Distances, And Bayesian Recognition

This section addresses the problem of variability in the polynomial coefficients with small changes in the data set by formulating it within a probabilistic framework. Constraining the polynomials to have bounded zero sets most often reduces the variability considerably. However, in some cases, the coefficients may still be sensitive to small changes in the data set. Also, unbounded polynomials are often more appropriate for representing portions of or all of an object as in the case with the handprinted characters shown in Figures 7(a)-(e). If the polynomial coefficients vary considerably, so will the invariants that are functions of these coefficients, thus giving unreliable recognition results. Thus, the first problem is to get an estimate of the variance of the polynomial coefficients. The second problem is to design a metric based on the polynomial coefficients for comparing two polynomial zero sets over the region where the data exists.

The input data here is a sequence of range data points, $\mathbf{Z}^N = \{Z_1, Z_2, \ldots, Z_N\}$, with $Z_i = (x_i, y_i, z_i)^t$.

Let $\boldsymbol{\alpha}$ denote the vector of coefficients of the polynomial $f(x, y, z)$ that describes the given object. We assume that the range data points $Z_1, Z_2, \ldots, Z_N$ are statistically independent, with Z_i having probability density function (pdf)

$$p(Z_i \mid \boldsymbol{\alpha}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2} \frac{f^2(Z_i)}{\| \nabla f(Z_i) \|^2}\right] \qquad (3)$$

The assumption is that Z_i is a noisy Gaussian measurement of the object boundary in the direction perpendicular to the boundary at its closest point. This model is introduced and discussed in [3, 18].

Thus, the joint probability of the data points is

$$p(\mathbf{Z}^N \mid \boldsymbol{\alpha}) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp[-\frac{1}{2\sigma^2} \sum_{i=1}^{N} \frac{f^2(Z_i)}{\| \nabla f(Z_i) \|^2}] \quad (4)$$

The maximum likelihood estimate $\hat{\boldsymbol{\alpha}}_N$ of $\boldsymbol{\alpha}$ given the data points is the value of $\boldsymbol{\alpha}$ that maximizes (4).

A very useful tool for solving the problems of object recognition and parameter estimation is an asymptotic approximation to the joint likelihood function, (4), which can be shown to have a Gaussian shape in $\boldsymbol{\alpha}$ [3], i.e.,

$$p(\mathbf{Z}^N \mid \boldsymbol{\alpha}) \approx [p(\mathbf{Z}^N \mid \hat{\boldsymbol{\alpha}}_N)] \exp\{-\frac{1}{2}(\boldsymbol{\alpha}-\hat{\boldsymbol{\alpha}}_N)\Psi_N(\boldsymbol{\alpha}-\hat{\boldsymbol{\alpha}}_N)\} \qquad (5)$$

where Ψ_N *is the second derivative matrix having* i,j*th* component, $-\frac{\partial^2}{\partial\alpha_i \partial\alpha_j} ln\, p(\mathbf{Z}^N \mid \boldsymbol{\alpha}) \mid_{\boldsymbol{\alpha}=\hat{\boldsymbol{\alpha}}_N}$. Hence, all the useful information about $\boldsymbol{\alpha}$ is summarized in the quadratic form in the exponent of equation (5). If Ψ_N is not singular, then it is the inverse covariance matrix of $\hat{\boldsymbol{\alpha}}_N$. Ψ_N is called the uncertainity matrix of $\hat{\boldsymbol{\alpha}}_N$.

The aposteriori distribution of $\boldsymbol{\alpha}$ given the data, i.e., $p(\boldsymbol{\alpha} \mid \mathbf{Z}^N)$, is propotional to $p(\mathbf{Z}^N \mid \boldsymbol{\alpha})p(\boldsymbol{\alpha})$. This can be written using the asymptotic approximation as

$$constant \times p(\mathbf{Z}^N \mid \hat{\boldsymbol{\alpha}}_N) \exp\left[-\tfrac{1}{2}(\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}_N)\Psi_N(\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}_N)\right] p(\boldsymbol{\alpha}) \qquad (6)$$

where $p(\boldsymbol{\alpha})$ is a prior distribution for $\boldsymbol{\alpha}$.

This distribution addresses the first problem because it tells us about the uncertainity in the polynomial coefficients given the data points. The uncertainity matrix Ψ_N defines an ellipsoid around $\hat{\boldsymbol{\alpha}}_N$ in the d-dimensional coefficient space. The axes of this ellipsoid are the directions of the eigenvectors of the uncertainity matrix, and the lengths of the axes are equal to the square roots of the eigenvalues. The volume of this ellipsoid gives a measure of the uncertainity in the parameter estimates. If the volume is large, then it implies that the coefficients are not reliable. If the coefficients are not reliable, neither will be the invariants that are functions of these coefficients. Then, instead of using the existing measurements to recognize the object, the system can collect more data in order to improve the parameter estimates (i.e., reduce the uncertainity volume). Details of how to collect more data in order to reduce the uncertainity volume as quickly as possible are given in [18].

4.1 Mahalanobis Distance as a Comparison Measure for Polynomial Zero Sets

The scenario that we consider here is one where we have a set of objects labeled $l = 1, 2, \ldots, L$ in the database. Each may be a polynomial of different degree n in x,y and z. Let $\boldsymbol{\alpha}^l$ be the parameter vector for object l. The optimum recognition rule is: 'choose l for which $p_l(\mathbf{Z}^N \mid \boldsymbol{\alpha}^l)$ is maximum'. This, however, requires considerable computation because the data is used L times to compute $p_l(\mathbf{Z}^N \mid \boldsymbol{\alpha}^l)$ for $l = 1, 2, \ldots, L$. However, if all the polynomials are of the same degree, considerable simplification results.

If all the objects are of the same degree, $p_l(\mathbf{Z}^N \mid \boldsymbol{\alpha}^l)$ will have the same form, $p(\mathbf{Z}^N \mid \boldsymbol{\alpha}^l)$, for all l, but will differ in the values of the parameter vector, $\boldsymbol{\alpha}^l$.

Using the asymptotic approximation, (5), we see that since $p(\mathbf{Z}^N \mid \hat{\boldsymbol{\alpha}}_N)$ is independent of l, an approximately equivalent recognition is : choose l for which (7) is minimum

$$(\boldsymbol{\alpha}^l - \hat{\boldsymbol{\alpha}}_N)^t \Psi_N (\boldsymbol{\alpha}^l - \hat{\boldsymbol{\alpha}}_N) \qquad (7)$$

The advantage in using (7) is that the data is involved just once (not L times) to compute the uncertainity matrix. Note that (7) is a Mahalanobis distance measure. The coefficient vector of the best fitting polynomial to the data is compared with the coefficient vector of each of the stored polynomials. To reiterate, the justification for using this distance measure is its equivalence to

checking how well the data set $Z_1, Z_2, \ldots, Z_N$ is fit by the polynomial having coefficient vector $\boldsymbol{\alpha}^l$.

An explanation for why the Mahalanobis distance is the appropriate metric for comparing polynomial zero sets is as follows. Assume that a polynomial of degree n is required to get a good fit to a bounded subset of data in the x-y plane. Since the data set is limited, it may not completely constrain the polynomial coefficients, thus permitting the polynomial coefficients to change greatly with small changes in the data set. Thus, even though two data sets may be very similar, the polynomials that fit them may look quite different far away from the data sets. In these circumstances, the coefficients of the polynomials that fit the data set and small variations of it, form a cloud of points in the coefficient space that is wide in some directions and narrow in others. A large variation therefore in the coefficients along the direction where the cloud is wide does not change the probability of the data very much. Thus, the direction of large cloud width is not very useful for comparing two polynomial zero sets. We, therefore, need a metric for comparing two polynomial zero sets that weights directions of large cloud width much less than directions of small cloud width. The Mahalanobis distance is such a metric.

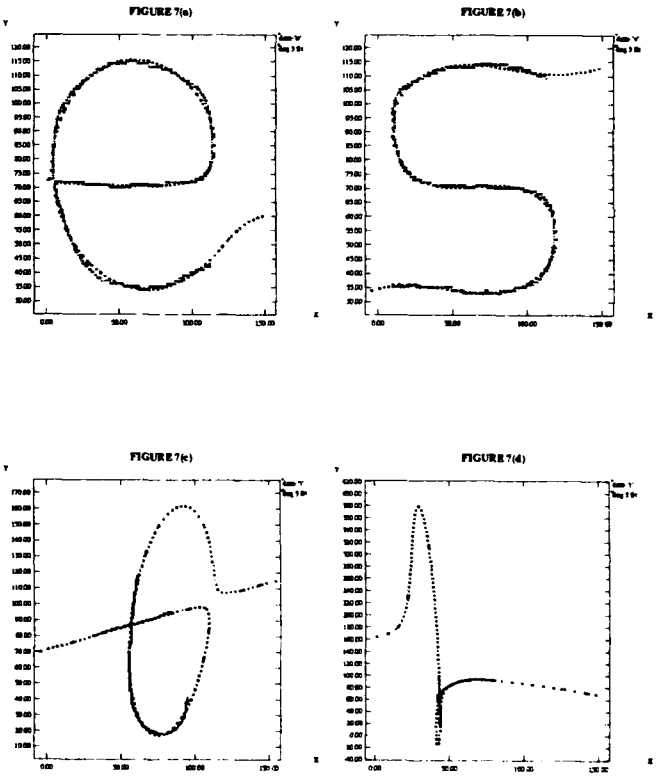
A direction of wide cloud width corresponds to an eigenvector of the uncertainity matrix for which the eigenvalue is small, and a direction of small cloud width corresponds to an eigenvector of the uncertainity matrix for which the eigenvalue is large. Thus, the Mahalanobis distance weights the directions of large cloud width much less than the directions of small cloud width. Infact, the Mahalanobis distance effectively uses a subspace of the coefficient space that is spanned by those eigenvectors of the uncertainity matrix corresponding to the eigenvalues that are larger than ϵ, where ϵ is small. Thus, the Mahalanobis distance is a metric that is useful for comparing two polynomial zero sets in the vicinity of the data set, based on the coefficient vectors.

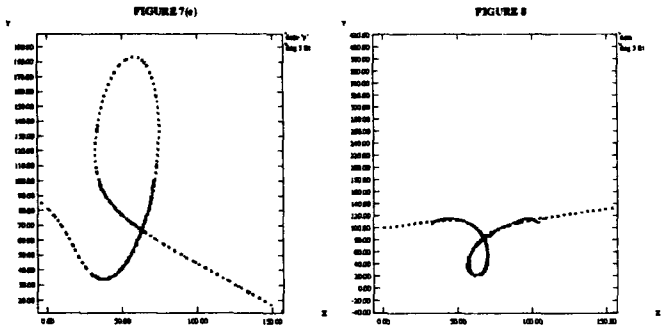
The scenario discussed here is one where each object in the database is characterized by one value of the parameter vector, $\boldsymbol{\alpha}^l$. A more general scenario is where each object has a parameter vector $\boldsymbol{\alpha}^l$ that is characterized by a distribution, $p(\boldsymbol{\alpha}^l)$. For example, a sphere with fixed center and radius that can take a range of values. [18] deals with computing a computationally attractive recognition rule for this general case.

4.2 Experimental Results

The set of results shown illustrates the use of the Mahalanobis distance for comparing polynomial zero sets. The data sets here correspond to handwritten characters and are all well fit by third degree polynomials. These are examples where unbounded polynomials are appropriate, because forcing the zero sets to be closed would require polynomials of higher degree, and in addition would make them look more like some of the other letters. The data sets in figures 7(a) through 7(e) are the objects in the database. These data sets are the letters, 'e','s','t','r' and 'y'. Also shown in these figures are the best third degree polynomial fits to the data. Figure 8 corresponds to another instance of the handwritten character 'r' that looks very much different from the one in the database. The Mahalanobis distance (7) of the coefficient vector for the best polynomial fit to this data set to the coefficient vectors for the best polynomial fits to the letters 'e','s','t', 'r' and 'y' in the database are 1.0e+5, 1.6e+5, 2.8e+4, 1.9e+3 and 8.8e+5 respectively. For computing the Mahalanobis distance, we scale all the data sets so that they all lie within a rectangle of the same dimension. Thus, all the data sets are compared on the same scale. From the results one can see that the distance is a minimum for the alphabet 'r'. Ofcourse, an 'r' like the one in the database would produce a much smaller distance.

The experiment, run on a SPARC 2, runs in close to real time. The polynomial fitting runs in real time. The computation of the second derivative matrix is fast because, due to the structure of the likelihood function, one can compute the second derivatives using a set of first derivatives. Thus, the Mahalanobis distance can be used to compare polynomial zero sets in real time.





References

[1] A. Albert. *Regression And The Moore-Penrose Pseudoinverse.* Academic Press, 1972.

[2] R. Bajcsy and F. Solina. Three-dimensional object representation revisited. In *International Conference on Computer Vision*, pages 231–240, London, May 1987.

[3] R.M. Bolle and D.B. Cooper. On optimally combining pieces of information, with applications to estimating 3D complex-object position from range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):619–638, September 1986.

[4] Bruno Cernuschi-Frias. *Orientation and Location Parameter Estimation of Quadric Surfaces in 3D from a Sequence of Images.* PhD thesis, Brown University, 1984.

[5] O.D Faugeras and M. Hebert. Segmentation of range data into planar and quadric patches. In *CVPR*, 1983.

[6] D.A Forsyth, J.L Mundy, , A. Zisserman, and C.M Brown. Projectively invariant representations using implicit algebraic curves. In *European Conference on Computer Vision*, June 1990.

[7] D.A Forsyth, J.L Mundy, , A. Zisserman, and C. Rothwell. Invariant descriptors for 3D recognition and pose. In *Application of Invariants in Computer Vision, Rikjavik, Iceland*, March 1991.

[8] J.H Grace and A. Young. *The Algebra of Invariants.* Cambridge University Press, 1903.

[9] A.D. Gross and T.E. Boult. Error of fit measures for recovering parametric solids. In *ICCV*, pages 690–694, 1988.

[10] C. Jerian and R. Jain. Polynomial methods for structure from motion. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12:1150–1167, December 1990.

[11] D. Keren, D. Cooper, and J. Subrahmonia. Describing complicated objects by implicit polynomials. Technical Report LEMS-93, Brown University, .1991.

[12] D.J. Kriegman and J. Ponce. On recognizing and positioning curved 3D objects from image contours. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12:1127–1138, December 1990.

[13] O.D.Faugeras, M.Hebert, and E.Pauchon. Segmentation of range data into planar and quadratic patches. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8–13, Washington DC, June 1983.

[14] V. Pratt. Direct least squares fitting of algebraic surfaces. *Computer Graphics*, 21:145–152, July 1987.

[15] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes.* Cambridge University Press, 1986.

[16] P.D Sampson. Fitting conic sections to scattered data. *Computer Vision, Graphics, and Image Processing*, 9:56–71, 1979.

[17] P.D Sampson. Fitting conic sections to very scattered data: an iterative improvement of the bookstein algorithm. *Computer Vision, Graphics, and Image Processing*, 18:97–108, 1982.

[18] J. Subrahmonia, D. Cooper, and D. Keren. Reliable object recognition using high dimensional implicit polynomials for 2D curves and 3D surfaces. Technical Report LEMS-94, Brown University, 1991.

[19] G. Taubin. 3D object recognition and positioning with algebraic invariants and covariants. IBM Technical Report RC-16211. Paper presented at the NSF Workshop on The Integration of Numerical and Symbolic Computing Methods, Saratoga Springs, July 1990, and is to appear in the book titled 'Symbolic and Numerical Computations-Towards Integration' published by Academic Press Limited.

[20] G. Taubin. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:1115—1138, November 1991.

[21] Gabriel Taubin. *Recognition and Positioning of Rigid Objects using Algebraic and Moment Invariants.* PhD thesis, Brown University, 1990.

[22] S. Wolfram. *Mathematica, a System for Doing Mathematics by Computer.* Addison-Wesley, 1988.

BUILDER-I: a System for the Extraction of SHGC Objects in an Edge Image

Hiroaki SATO* and **Thomas O. BINFORD**
Robotics Laboratory
Department of Computer Science
Stanford University, CA94305

Abstract

A set of modules to extract partial descriptions of SHGC objects in an edge image is presented. It consists of modules to find end edges, to find meridian edges, to find cross-section edges, and to recover 3D shapes. The first goal of the system is to extract geometrical edges derived from an SHGC object and the second goal is to recover 3D information of the object. From an input edge image, pairs of end edges are detected first by verifying strong geometrical constraints for the ends of an SHGC, then meridian edges are detected by using the constraint for tangent intersections and the ones related to the end edges. Further, the axis of SHGC and the axes of skewed symmetry in cross-section edges are detected. And finally original cross-section and the sweeping rule are recovered by utilizing these three orthogonal axes. Extracted geometrical edges and 3D information from real images are shown.

1 Introduction

Though the recovery of 3D shapes from an image is in general underconstrained, it is simplified when objects belong to a generic object class. The class of generalized cylinders [Binford 71] [Agin and Binford 73] is a popular representation scheme for curved objects and the class of straight homogeneous generalized cylinders (SHGCs) is a subset of generalized cylinders where the objects are obtained by sweeping arbitrary cross-section with arbitrary scaling along a straight axis [Shafer and Kanade 83] [Ponce et al. 89] (figure 1).

In this paper, we address the problem to extract partial descriptions of SHGC objects from a real image by *constraint-based edge grouping*, i.e., by using geometrical constraints derived from restrictions in shape models. We have developed a set of modules called BUILDER-I (Bottom-Up Image Level Description ExtractoR - version I) which consists of a module to find end edges of an SHGC, one to find meridian edges, one to find cross-section edges, and one to recover 3D shapes from the meridian and cross-section edges. The importance of finding meridians and ends is two-fold: in the context of model-based vision systems, it makes hypothesis generation for objects in the image feasible by detecting a limited number of edge sets for an object part; also it makes methodologies to recover 3D shapes of generalized cylinders developed in this field be applicable to real images.

Though researchers showed that the 3D shapes of an SHGC is recoverable from its cross-section and limbs or meridians with a few degree of freedom, and that various useful invariants exist for the contours of an SHGC [Rao and Medioni 88], only little work has been done for finding geometrical edges in a real image except finding some type of ribbons or symmetries, e.g., [Brady and Asada 84], [Ponce 90], or [Sumanaweera et al. 88].

Mohan and Nevatia proposed a method of perceptual organization for segmentation and description from intensity edge contours. It performs a similar type of task as BUILDER-I not by applying geometrical constraints but by using a neural network for grouping consistent symmetries [Mohan and Nevatia 89].

Saint-Marc and Medioni proposed a method to find skewed and parallel symmetry in a real image [Saint-Marc and Medioni 90]. Parallel symmetry is closely related to the ends of SHGC. However, it is expected that their method is not suitable to extract the end edges because parallel symmetry is more general relation of curves than the ends of SHGC and it lacks some of the geometric constraints required for the ends of an SHGC.

Ponce et al. showed algorithms to detect axes of SHGCs in an edge image [Ponce et al. 89]. The same approach as ours was used in their work, i.e., clarifying the geometrical constraints derived from the shape restrictions on SHGCs mathematically, then utilizing them to develop algorithms to extract partial descriptions of SHGC objects. However, the algorithms have unnegligible restrictions to object shapes: an implicit restriction in the first algorithm derived from Hough transform [Hough 62] is that the object needs to have sufficient number of edgels for grouping, i.e., long limbs or many meridians; the restriction in the second algorithm is the number of zero-curvature points in the limbs as the authors noted.

We developed algorithms to find the ends of SHGC

*Information Systems Research Center, Canon Inc., 890-12 Kashimada, Saiwai-ku, Kawasaki, Kanagawa 211, Japan. email: sato@cis.canon.co.jp

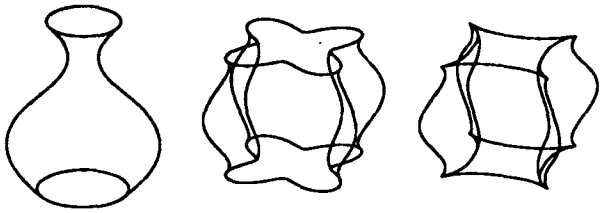


Figure 1: Straight homogeneous generalized cylinders.

in an edge image since strong geometrical constraints exist for them [Sato and Binford 92]. Though they assumes additional restrictions to object shapes or a priori information to reduce computing time, the method itself is applicable to the no a priori information and no additional restriction case. The algorithms showed the ability to find pairs of end edges with a limited number of false positives. And it is expected that detection of meridian edges can be more reliable and efficient by combining the information from the end edges.

The recovery of 3D shape of generalized cylinders has been tackled. To fix the remaining freedom, certain additional constraints, e.g., constraints from shading or heuristic ones like orthogonality constraints, are introduced [Gross and Boult 90] [Ulupinar and Nevatia 90]. In this paper, additional explicit restrictions to object shapes, i.e., RSHGC with symmetric cross-section, are introduced. It is applicable only to a subset of SHGC objects, however, the geometrical constraints derived from them make the algorithm simple and efficient with only the assumption of skewed symmetry. The rest of the paper is organized as follows:

In section 2, overview of the BUIDER-I system is presented. In section 3, the algorithms to find the ends of SHGCs is summarized briefly. In section 4, the algorithms to find meridian edges and cross-section edges are presented. Geometrical constraints used in the algorithms and techniques to utilize them are described. In section 5, a method to recover 3D shapes of an SHGC is presented. Geometrical relation between the projections of orthogonal axes and viewing transform is clarified and a simple method to find axes of skewed symmetry is described. Experimental results from real images are shown throughout the paper.

2 Overview of BUILDER-I system

The task of BUILDER-I is the extraction of partial descriptions for objects in an image. BUILDER-I is designed to be a subsystem in a flexible model-based vision system, such as ACRONYM [Brooks 81] and SUCCESSOR [Binford et al. 87], which combines bottom-up and top-down processing. The descriptions from BUILDER-I are supposed to be used by top-down modules to index appropriate models and to predict the appearance of objects in images.

The input to BUILDER-I is an edge image in which position and orientation of edgels are measured in reasonable accuracy and the edgels are linked / divided with their continuity. BUILDER-I consists of four modules:

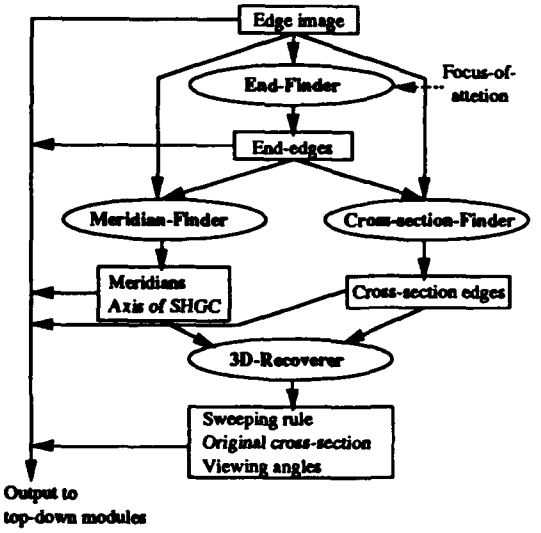


Figure 2: BUILDER-I system configuration.

one for finding the ends of SHGCs, named *end-finder*, one for finding meridian edges, *meridian-finder*, one for finding cross-section edges, *cross-section finder*, and one for recovering sweeping rules and original cross-sections, *3D-recoverer*. Configuration of BUILDER-I system is shown in figure 2. Though the initial step to extract visual clues from an image can be either finding ends or finding limbs / meridians [Rao and Nevatia 87], the former is selected because strong geometrical constraints exist for the ends of an SHGC. Currently, the system is not fully automatic: an edge for focus of attention, actually one of the end edges, has to be indicated manually in order to reduce computing time (see section 3).

The behavior of the system is as follows: first, the ends of an SHGC are detected by end-finder from the input edge image with a given reference edge; then, meridian edges are detected by meridian-finder using the ends detected in the first step; also, cross-section edges corresponding to the ends are detected by cross-section-finder; finally, if the meridian and cross-section edges are found and if the object shape can be assumed to be a RSHGC with its axis passing through the center of its cross-section, 3D-recoverer is activated, and if the cross-section has the axes of skewed symmetry, the original cross-section and the sweeping rule are recovered. The results of the modules, i.e., the ends, the meridian edges and their axis, the cross-section edges, the sweeping rule, the original cross-section, and the viewing angles are combined to output.

3 Algorithms to find the ends of an SHGC

In this section, we briefly summarize the algorithms to find the ends of an SHGC because it is an important module in BUILDER-I. More precise description of the algorithms can be found in [Sato and Binford 92].

In this paper, we use the terms, such as *SHGC*,

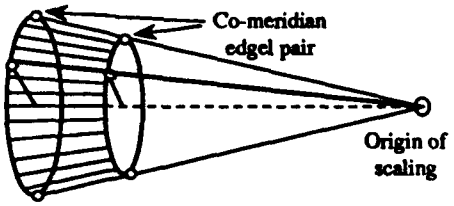


Figure 3: Origin of scaling and co-meridian edgel pair. Corresponding co-meridian edgel pairs are shown with the lines connecting each edgel pair.

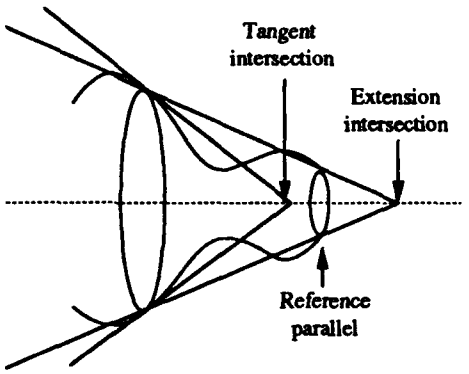


Figure 4: Tangent intersection and extension intersection.

RSHGC, LSHGC, meridian, parallel, and limb, according to [Shafer and Kanade 83] and [Ponce et al. 89]. Since the primary task of BUILDER-I is 2D image analysis, those terms are often used for the projections of objects to 2D images.

The term *cross-section edges* are used for edges at the intersections of a side face and end faces, i.e., end parallels. On the other hand, *original cross-section* is used for a cross-section function which is used to generate the object with a sweeping rule. *Ends of an SHGC, end edges, or ends*, are used for a part of cross-section edges which bounds a visible part of the side face. *Co-meridian* (respectively *co-parallel*) edgels or co-meridian edgel pair is used for a pair of edgels in the ends (meridians) which lie in a meridian (parallel). (Figure 3.) *Corresponding co-meridian* (respectively *corresponding co-parallel*) edgel pairs is used for co-meridian edgel pairs whose edgels lie in the same pair of parallels (meridians). *Tangent intersection* is used for a point at which tangent lines to limbs or meridians at each edgels in a co-parallel edgel pair intersect. *Extension intersection* is used for an intersection point made by a pair of extension lines which pass through each edgel pair in corresponding co-meridian edgel pairs (figure 4). *Origin of scaling* is used for a point which is the extension intersection for the ends of an SHGC. We use the term origin of scaling rather than the term *apex of shape* because apex of shape seems to be used for a virtual apex of an LSHGC

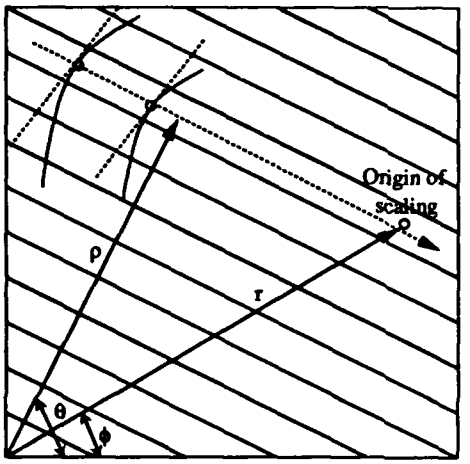


Figure 5: Projection and position of origin of scaling.

in [Shafer and Kanade 83]. *Scaling ratio* is used for the ratio of scaling factors for a pair of parallels . The imaging process is assumed to be orthographic projection in this paper.

The constraints used to detect the ends of an SHGC are as follows:

End-constraint(i) The edgels which form a co-meridian edgel pair have the same orientation.

End-constraint(ii) The extension intersections for corresponding co-meridian edgel pairs coincide at a point in the axis.

End-constraint(iii) The scaling ratio for each edgel pair in corresponding co-meridian edgel pairs is the same.

End-constraint(iv) The vector between each edgel pair in corresponding co-meridian edgel pairs is the same when its scaling ratio is equal to 1.

Because a reliable and efficient pre-processing for curve fitting or selection of feature points is another difficult problem, the algorithm shown below is a simple edgel-based method, i.e., it selects and groups pairs of edgels which satisfy the above constraints. Without accurate measurement of curvature at edgels, it is required to hypothesize the position of origin of scalings. Thus, the computational complexity of the algorithm becomes $O(nkdc)$ by using *projection* technique in [Nevatia and Binford 77], where n is the number of edgels, k is the number of discretized projection directions, d is the number of hypotheses for an edgel pair, c is the number of partially parallel edges in an image. Unfortunately this would require very long computing time and we introduced additional constraints to avoid it. The first algorithm is for cylindrical ends, which means scaling factors at the ends are the same. The second algorithm is for any SHGC. However, a modified version is implemented; given a reference end edge, it finds edges possibly paired with it. The second algorithm is summarized as follows (see figure 5 also):

1. discretize orientation for projection direction, discretize two-dimensional space for origin of scaling, and prepare buckets for projection and cells for origin of scaling,
2. group the edgels according to their orientation, (the results are called orientation-groups,)
3. for each projection direction, and for each orientation-group,
   (a) project each edgel in the group into a bucket, (the results are called projection-orientation-groups,)
   (b) calculate extension lines in the projection direction and determine the cells (r, ϕ) which each extension passes through,
   (c) put the projection-orientation-groups to the cells,
4. select candidates for the cell position for origin of scaling
   (more precisely in [Sato and Binford 92]).
5. in each candidate cell (r, ϕ),
   (a) for each projection-orientation-groups in the cell, generate pairs of edgels in the group which consist of an edgel in the reference edge and another in other edges, then calculate the scaling ratio for each pair,
   (b) group the edgel-pairs according to their scaling-ratios and position of the cell, (the results are called scaling-origin-groups,)
6. for each scaling-origin-group, restore the edge-pairs corresponding to the edgel-pairs in the group.

Figure 6 and figure 7 show the extracted end pairs by the implemented algorithms. The algorithms detect most of the possible end pairs and few false positives because the constraints collected along an edge are strong enough to exclude most false positives. Also, except for the computing time, the method used here is applicable in the case with no additional restriction and no a priori information simply by giving each edge as the reference edge.

4 Finding the meridian edges of an SHGC

4.1 Overview of the algorithm to find meridians

It is shown by Shafer and Kanade that the tangent lines to meridians at each edgel in a co-parallel edgel pair intersect at a point in the axis. Also other researchers showed that tangent lines to limbs at for a co-parallel edgel pair intersect at a point in the axis in the imaging plane [Ponce et al. 89] [Ulupinar and Nevatia 90]. We call those properties the constraint for tangent intersections. The constraint for tangent intersections can be used to find the axes of SHGCs, and its simplest utilization is Ponce's $O(n^2d)$ algorithm.

Unfortunately, the constraint for tangent intersections only is too weak to prune candidates for co-parallel edgel pairs directly, so that the implicit restriction derived from Hough Transform was introduced. In this paper, we also use the following constraints which are related to the junction points between the ends and meridians.

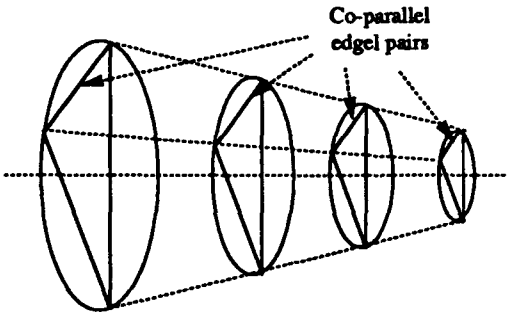


Figure 8: Corresponding co-parallel edgel pairs.

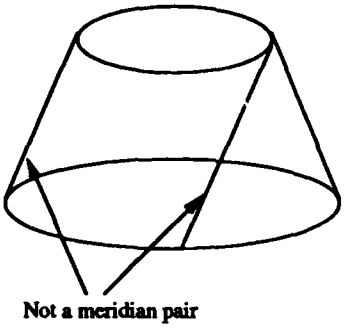


Figure 9: False correspondence for meridians.

Constraint(i) All vectors between each edgel pair in corresponding co-parallel edgel pairs are parallel (figure 8).

Constraint(ii) For a pair of parallel, extension intersections are the same point in the axis.

The proofs for these properties are given in appendix.

With one of the ends, co-parallel edgel pairs can be determined directly by the constraint (i). Still, axes are not constrained sufficiently, so that the algorithm might find wrong correspondences (figure 9). By using end pairs, those wrong correspondences are rejected, so that we consider the only meridians that connect two ends in the following algorithm.

Another problem in finding meridians is that the constraints do not determine if a single edge path is a meridian or not, but find that a pair of edge paths or two pairs of edge paths are not consistent as meridians. Inconsistent-sets generated by applying the constraints are used to find groups of edge paths as candidates for consistent meridians.

The algorithm is summarized as follows:

1. find junctions between an end edge and other edges.

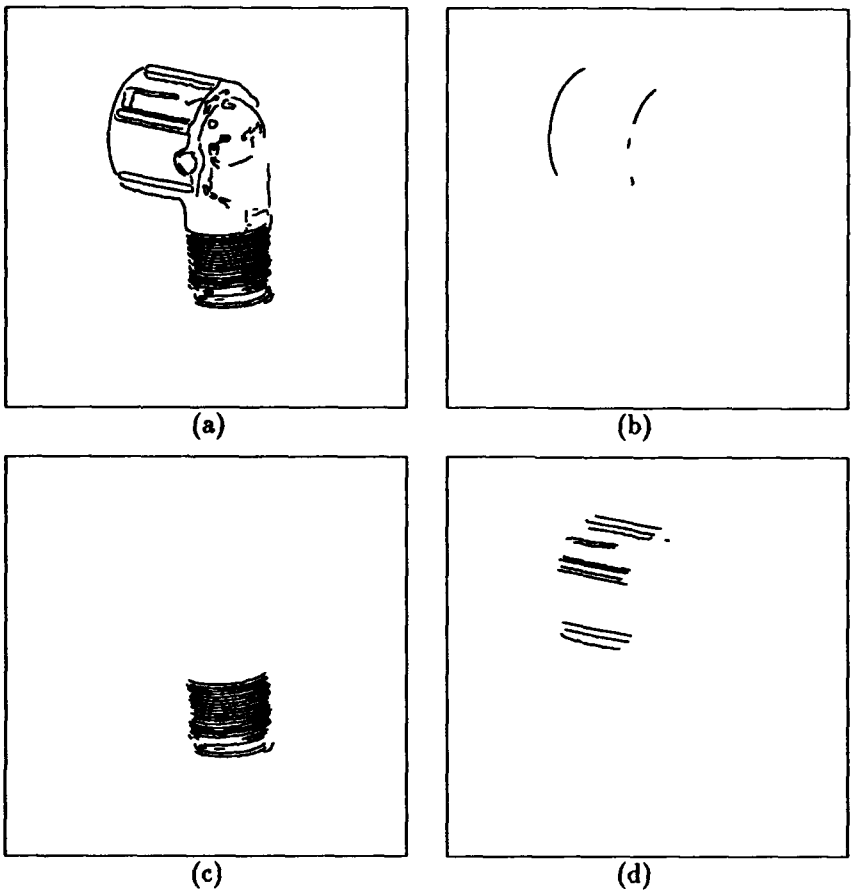


Figure 6: Examples of the results in end-finder. (a): input edge image, (b)(c)(d): extracted end edges.

2. select pairs of junctions which could be the ends of a meridian.
3. find edge paths which connect each pair of junctions.
4. repeat the followings until meridians are found.
   (a) select the edge path that connect the largest number of parallels in untested edge paths.
   (b) select edge paths which connect the parallels that terminate the edge path in step (a).
   (c) for each pair of selected edge paths, calculate their corresponding angle, corresponding co-parallel edgels, scaling ratios, tangent intersections, and extension intersections.
   (d) verify if the scaling ratios coincide or not for two pairs of edge paths.
   (e) verify if the tangent intersections coincide or not for two pairs of edge paths.
   (f) generate consistent-sets for the inconsistent-sets in step (c), (d), and (e).
   (g) for each consistent-set, verify if the tangent and extension intersections lie in a line or not by calculating normalized least squares line.

4.2 Techniques to find meridian edges

4.2.1 Finding junctions between the ends and other edges

To find the junctions, it is necessary to infer missing parts of edges. We assume that extrapolations of edges can be used to approximate the missing parts. Values of curvature are determined by fitting a linear function to measured orientation values near the ends of edges. By assuming a constant curvature, the extrapolations of edges are calculated. And if they come sufficiently close, the junctions are hypothesized.

Figure 10 shows hypothesized junctions. Filling the missing parts by extrapolating usually gives sufficient approximation, however, if the change in curvature is not small, e.g. around a limb, the error in the position and orientation can be fairly large. This makes it impossible to use very strict conditions for finding junctions and the algorithm may find many false junctions as well as the right ones.

4.2.2 Finding corresponding junction pairs

In the next stage, pairs of junctions which could be the ends of a meridian are selected. Constraints to a junction pair are as follows:

- orientation of the tangents to end edges at the ends of a meridian are the same.

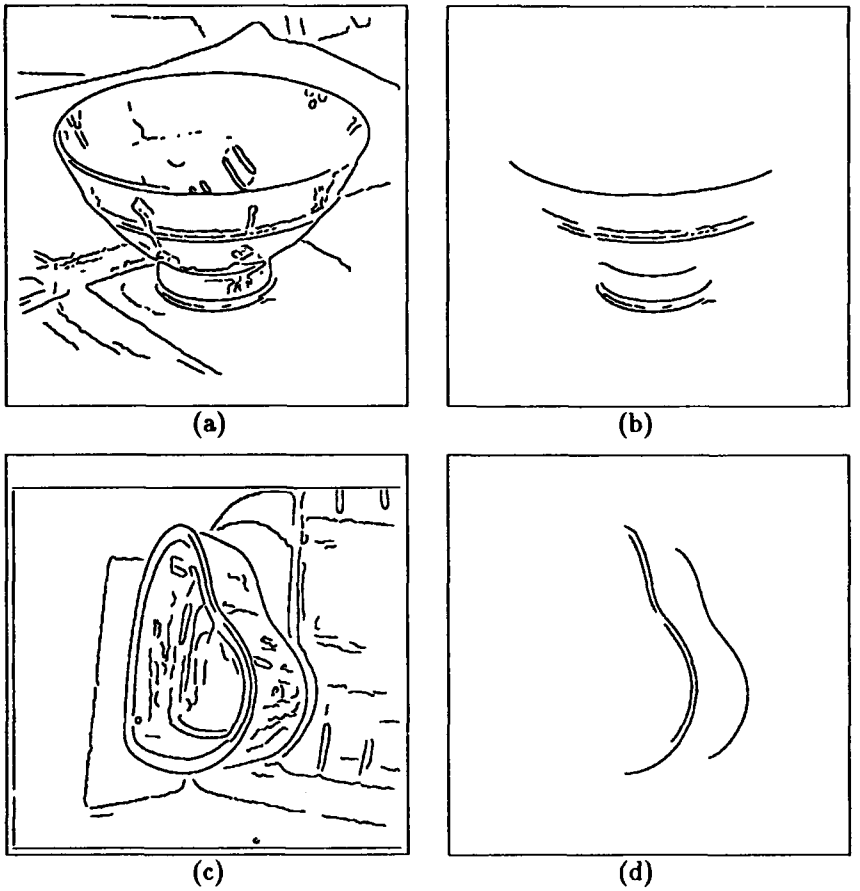


Figure 7: Other examples of the results in end-finder. (a)(c): input edge images, (b)(d): extracted end edges.

- extension of the line passing through the ends of a meridian passes by the origin of scaling.

A polar coordinates with non-linear scaling in [Sato and Binford 92] is used to verify if the extension lines pass by the origin of scalings. Figure 11 shows the junction pairs extracted by the algorithm with the lines which connect each junction pair.

4.2.3 Searching edge paths between a junction pair

Edge paths which connect a junction pair are detected as candidates for meridians in the next stage. Edge links are found by putting a rectangular box around the ends of edges. Best first search algorithm is used to find edge paths; starting from an edge at a junction, repetitively, the edge path that has the minimum length in intermediate edge paths is selected and extended with the edge links from its end until it reaches the other junction or exceeds a predetermined length.

4.2.4 Finding corresponding co-parallel edgel pairs

Corresponding co-parallel edgel pairs can be detected by applying the constraint (i). For a pair of edge paths, the correspondence angle between a co-parallel edgel pair is calculated from the junctions in an end. And by using projection in that angle, corresponding co-parallel edgel pairs are found. Figure 12 shows examples of corresponding co-parallel edgel pairs. During the projection the following constraints are verified: projected positions of edgels along a meridian must be monotonic because a sweeping rule is a single-valued function along Z axis (figure 13); a pair of meridians must not share an edge segment. If these constraints are not satisfied, inconsistent-sets are generated since the pair of edge paths are not consistent as meridians.

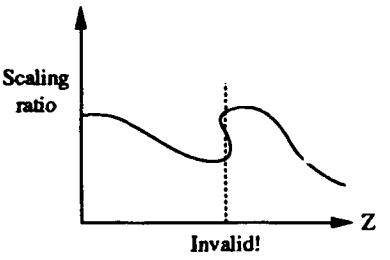


Figure 13: Invalid sweeping rule.

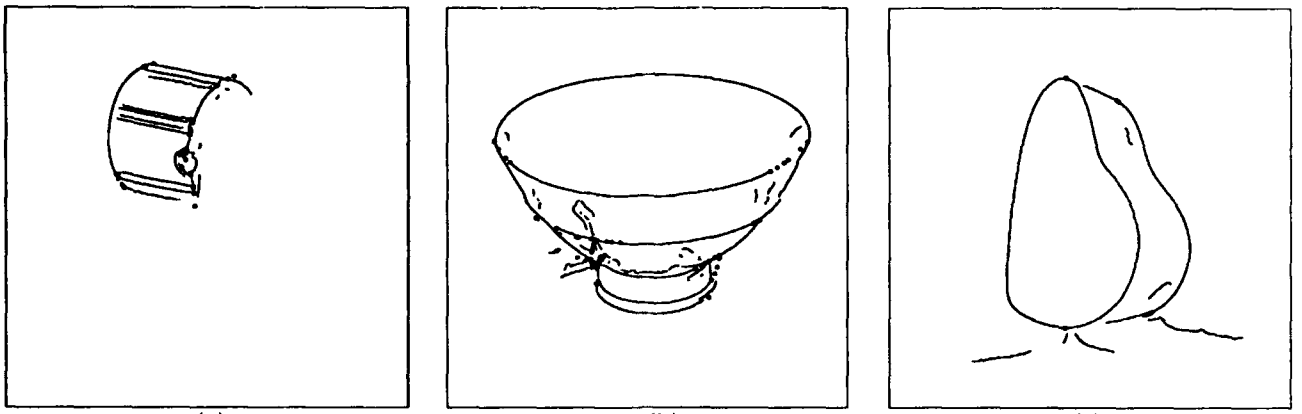


Figure 10: Hypothsized junctions from the ends.

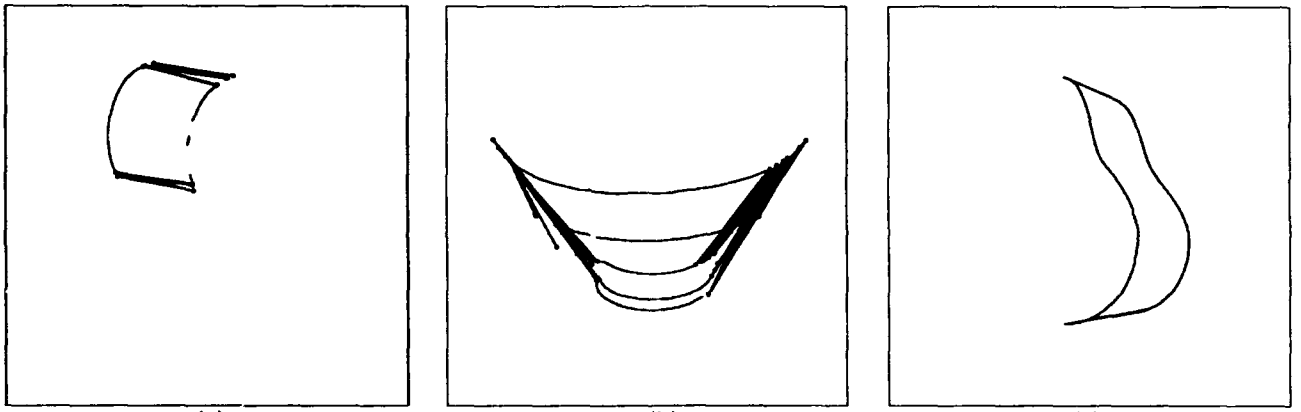


Figure 11: Junction-pairs for the ends of meridians.

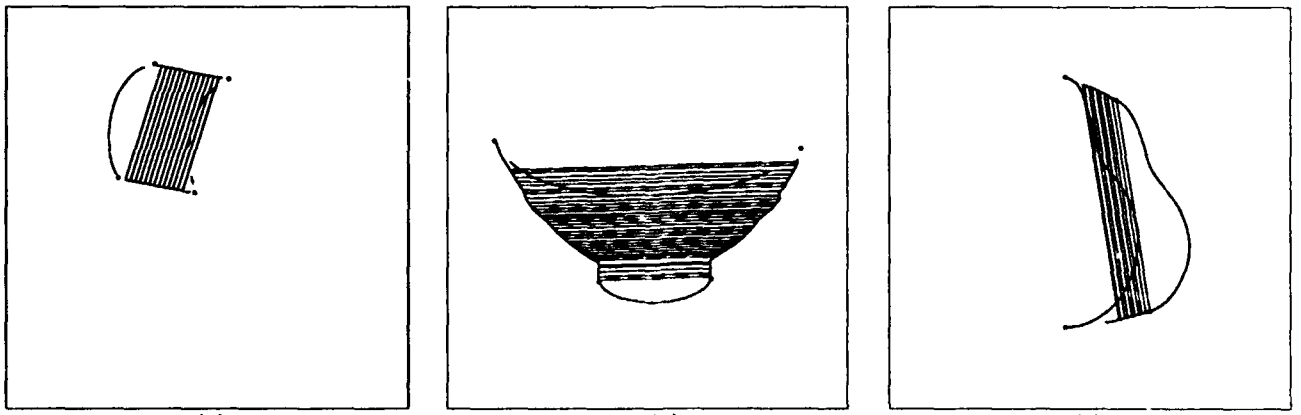


Figure 12: Examples of corresponding co-parallel edgel pairs.

4.2.5 Inconsistent-sets by scaling ratios and tangent intersections

Each corresponding co-parallel edgel pair gives a scaling ratio and a tangent intersection for a parallel, though its position along the axis is not determined yet. Two pairs of edge paths give two sweeping rules and two sets of tangent intersections that should coincide. Averaged difference of sweeping rules, as well as tangent intersections, is calculated for two pairs of edge path that share an edge path, namely (M1, M2) and (M1, M3). If the difference is not sufficiently small, an inconsistent-set of three edge paths, (M1, M2, M3) is generated. To calculate distance between tangent intersections, a polar coordinate system with non-linear scaling is used for a rough approximation of error normalization as in section4.2.2.

4.2.6 Consistent-set generation for a group of meridians

The constraints yield inconsistent-sets as described above. A group of meridians that satisfy a consistent interpretation must not be a superset of any inconsistent-sets. Manipulation of consistent-sets can be implemented efficiently with simple boolean operations on bit vectors as used in ATMS. For instance, the condition, A is a superset of B, is $\sim B \oplus B \otimes A = T$, where $\sim$, $\oplus$ and $\otimes$ are bit-not, bit-or and bit-and operator, and each bit in the bit vectors, A and B, indicates that each hypothesis is true or false, i.e., each edge path is a meridian or not. By generating combinations of hypotheses and pruning them with the inconsistent-sets, consistent-sets are acquired.

Though this method might take long time for a large number of hypotheses, it is useful when the number of hypotheses are limited, e.g. to 20 or 30, and many of the hypotheses are pruned in the early stages.

4.2.7 Normalized least squares method to find the axis of SHGC

The final condition to verify for a meridian group is that their tangent intersections and extension intersections lie in a straight line. Least squares method is used to calculate the best fit line and the error from it.

The method used here normalizes the errors according to the following model (figure 14): For extension intersections, the error in the position of an edgel P, is magnified by $(l + l_0)/l_0$ to the error in extension intersection e, where l_0 is the distance between the junction and P; l is the distance between P and the extension intersection. Thus, the normalized error $\bar{e}$ is:

$$\bar{e} = e \cdot l_0/(l + l_0) = e \cdot \frac{s_0 - s_1}{s_0},$$

where s_0 and s_1 are the distances between co-parallel edgel pairs. The normalized least squares method minimizes not $\sum e^2$ but $\sum \bar{e}^2$. In the same manner, the normalized error $\bar{e}$ for a tangent intersection is approximated to:

$$\bar{e} = e\frac{l_0 \cdot |\cos(\phi - \theta_2) - \cos(\phi - \theta_1)|}{s_0},$$

where ϕ is the angle between a co-parallel edgel pair; θ_1 and θ_2 are the orientation at the edgels; l_0 is the length of the edge curve used to calculate the orientation; s_0 is the distance between the edgel pair.

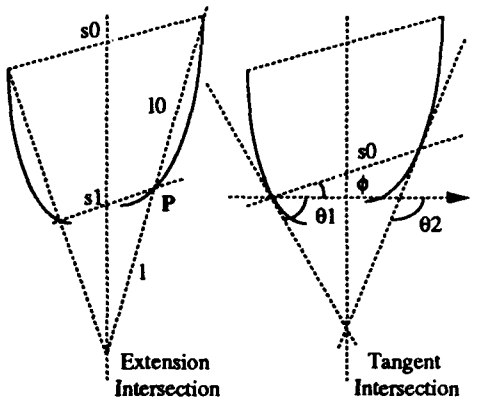


Figure 14: Error model for intersections.

Figure 15 shows examples of detected meridians and axes by the algorithm.

In figure 15 (a), only the meridian edges which have junctions at both ends are detected. In (b), the limbs are detected though they are not meridians strictly, because the condition (i) is satistied when the object is a solid of revolution. In (c), the limbs are detected because limbs by a linear sweeping rule are meridians.

Table 1 shows averaged square errors by the normalized least squares method and usual least squares method. The errors by the normalized least squares method is stable and interpretable, i.e., the errors in position at edgels are less than 1 (pixel) when the intersections lie in a line. (The maximum error in figure 15 (a) is quite large because many of the intersections are nearly plus and minus infinite. On the other hand, the errors by usual least squares method are magnified by the extension length, so that they are not suitable to verify if the intersections are in a line or not.

The cases are further divided into where tangent and extension intersections are distributed in a line, where they are at a finite focal point, or where they are at an infinite focal point. These three correspond to a non-linear sweeping rule, a linear (not constant) sweeping rule, and a constant sweeping rule respectively. The maximum errors and ratio of infinite foci are used to determine if the meridians have an axis or a focal point. Actually, the axes are not fully constrained in case (a) and (c). The axes in the figures are drawn as they pass through the center of cross-section or the center of the given end edge to show the acquired results.

In general, limbs are not meridians and an object doesn't necessarily have two or more meridians. Fortunately, our method is extensible to limbs when a corresponding cross-section is detectable, though this extended version is not implemented yet. (See next section about detection of cross-section.) It is well-known that orientation of the tangent to limbs and that to the corresponding point in the cross-section are the same in the imaging plane (figure 16), so that the corresponding

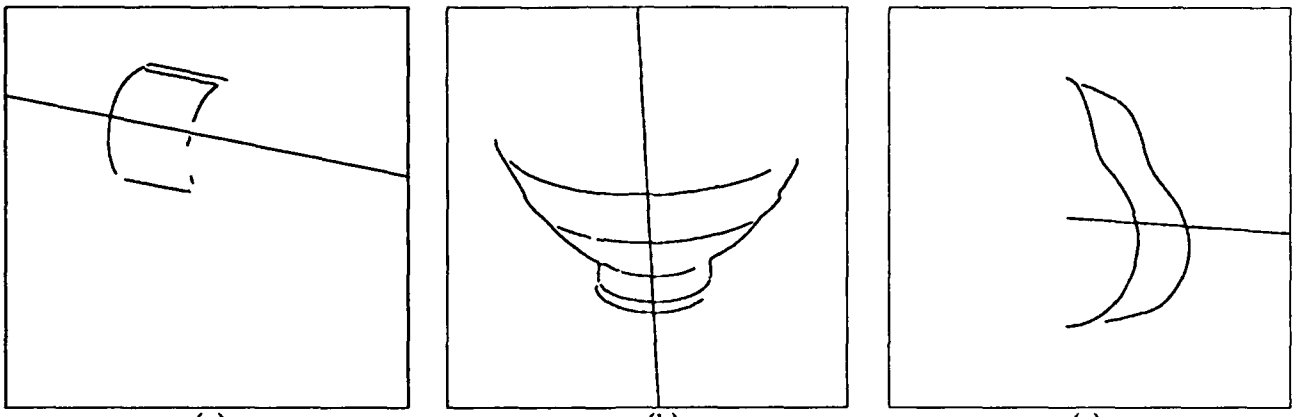


Figure 15: Detected meridian edges and axes.

method	normalized least squares		usual least squares		infinite foci
data	min error	max error	min error	max error	ratio
ElbowJoint: figure15(a)	0.08952	537.90	9081.7	6.2926E6	0.580
RiceBowl: figure15(b)	0.13935	11.730	861.24	1.6723E5	0.045
HeartCup: figure15(c)	0.02379	0.22385	371.24	2171.2	0.0

Table 1: Averaged squares error by normalized least squares fitting to a line and ratio of infinite distant intersections.

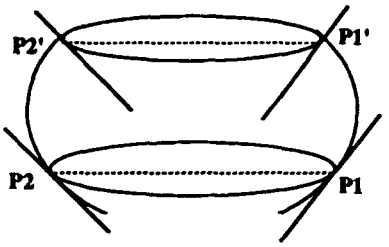


Figure 16: Relation between limb points and cross-section points.

points P' in the cross-section can be determined from a limb point P. If the angle between two limb points $P1$ and $P2$ is equal to the angle between the corresponding points in cross-section $P1'$ and $P2'$, they are supposed to lie in a parallel. The same constraint is used to find parallels from a given cross-section and limbs in the previous work.

Performance of the algorithm, both its detectability and computing time, depend on the ability to find junctions correctly. The algorithm to find junctions used here is not so reliable that it detects many false junctions and causes long computing time. Once junctions are found, the constraints are strict enough to reject false meridians in our experiments.

4.3 Finding cross-section edges in an edge image

The module to find cross-section edges is a necessary part of the program. We describe the algorithm to find cross-section edges in this section, though it has an obvious limitation and should be replaced by another algorithm using more sophisticated constraints in the future.

If there is no restrictions to cross-section shape in three space, no strong geometrical constraints is available to detect projected cross-section edges. Thus we impose an additional restriction to cross-section shapes, i.e., the cross-section should be a C^1 curve.

Currently, the algorithm finds the minimum length C^1 closed edge path. The algorithm is a best-first search starting at the end edges detected by the end-finder module. The shortest edge path is repetitively selected and updated with C^1 edge links found from its end until a closed path is found or the length of path exceeds a certain threshold. The method to find C^1 edge links is similar to the one for finding junctions. The difference is that the former verifies difference of orientations between two edges at the edge link.

Detected cross-sections are shown in figure 17. This algorithm has the same weak point as the junction finding algorithm in section 4.2.1. Still, it can detect correct cross-sections mostly when no significant obscurations or missing parts exist in the cross-section.

5 Recovering 3D shape of SHGC

Though the geometrical constraints for contours only are not sufficient to recover the 3D shapes of SHGCs, it is possible when some more constraints are available. Since a planar region, the projected shape of the cross-section, is detected by the system, skewed symmetry gives an important constraint as in the analysis of polyhedral objects. Two more additional constraints are derived from assumptions on object shapes:

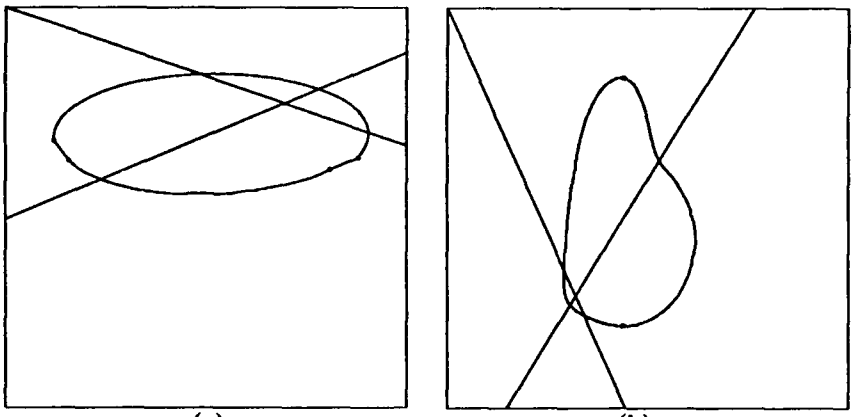


Figure 17: Detected cross-section edges and axes of skewed symmetry.

- The object belongs to Right SHGCs.
- The axis passes through the center of cross-section.

With these assumptions, the detection of three orthogonal axes in an image becomes possible, and we have sufficient constraints to recover 3D shape quantitatively.

5.1 Recovery of 3D information from three orthogonal axes

In this section, we show the relation between the projection of three orthogonal axes and viewing angles, and then the relation of object contours in an image to its sweeping rule and original cross-section.

According to [Ponce et al. 89], the contours of RSHGC in the imaging plane are:

$$\begin{aligned}\overrightarrow{OQ} &= \rho(\theta)r(z)\sin(\theta-\alpha)\vec{w} \\ &\quad +[z\sin\beta-\rho(\theta)r(z)\cos\beta\cos(\theta-\alpha)]\vec{u}.\end{aligned}$$

where $\overrightarrow{OQ}$ indicates the position in the image; $(\vec{w},\vec{u})$ is unit bases in the imaging plane; (α,β) is the viewing angle (figure 18).

By substituting the values for θ and z, we get the projection of three axes as:

$$\overrightarrow{OQ_x} = -\rho(0)r(0)(\sin\alpha\vec{w}+\cos\beta\cos\alpha\vec{u}).$$

$$\overrightarrow{OQ_y} = \rho(\frac{\pi}{2})r(0)(\cos\alpha\vec{w}-\cos\beta\sin\alpha\vec{u}).$$

$$\overrightarrow{OQ_z} = z\sin\beta\vec{u}.$$

An actual image is obtained by rotating and transfering the projected object contours in the imaging plane. By letting the angles of axes before rotation be ψ_x, ψ_y, and ψ_z, and the angles after rotation be ψ'_x, ψ'_y, and ψ'_z, we get the following relations:

$$\psi'_x = \psi_x+\gamma,\ \psi'_y = \psi_y+\gamma,\ \psi'_z = \psi_z+\gamma.$$

$$\tan\psi_x = \frac{\cos\beta\cos\alpha}{\sin\alpha},\ \tan\psi_y = -\frac{\cos\beta\sin\alpha}{\cos\alpha}.$$

where γ is the rotation angle around viewing axis. Since ψ_z is $\pi/2$, we get the rotation angle $\gamma = \psi'_z - \pi/2$.

From the last two equations, we get the following relations:

$$\cos\beta = \pm\sqrt{-\tan\psi_x\tan\psi_y},$$

$$\alpha = \mp\arctan\sqrt{-\frac{\tan\psi_y}{\tan\psi_x}}.$$

The tansfer vector in the imaging plane is determined from the center of the cross-section because the axis of SHGC is assumed to pass through it. After cancelling the rotation and transfer, projected cross-section is observed as:

$$w = \rho(\theta)r(0)\sin(\theta-\alpha),\ u = -\rho(\theta)r(0)\cos\beta\cos(\theta-\alpha).$$

Thus, by using (w,u) coordinate system and substituting α and $\cos\beta$, original cross-sections are calculated as:

$$\rho(\theta)\cdot r(0) = \sqrt{w^2-\frac{u^2}{\tan\psi_x\tan\psi_y}},$$

$$\theta = \pm\arctan\frac{w\cdot\sqrt{-\tan\psi_x\tan\psi_y}}{u} \mp \arctan\sqrt{-\frac{\tan\psi_y}{\tan\psi_x}}.$$

Also, points along a projected meridian, i.e.,points by changing z value with a constant θ, are:

$$w = \rho(\theta)r(z)\sin(\theta-\alpha),$$
$$u = z\sin\beta-\rho(\theta)r(z)\cos\beta\cos(\theta-\alpha).$$

The junction between an end $(z=0)$ and a meridian (w_0,u_0) is:

$$w_0 = \rho(\theta)r(0)\sin(\theta-\alpha),$$
$$u_0 = -\rho(\theta)r(0)\cos\beta\cos(\theta-\alpha).$$

By using (w,u) coordinate system and (w_0,u_0), sweeping rules, $R(z)=r(z)/r(0)$, are calculated in:

$$R(z) = \frac{w}{w_0},\ z = \pm\frac{1}{\sqrt{1+\tan\psi_x\tan\psi_y}}(u-\frac{w}{w_0}u_0).$$

Signs give reverse or symmetric shapes since the direction of axes may be reversed.

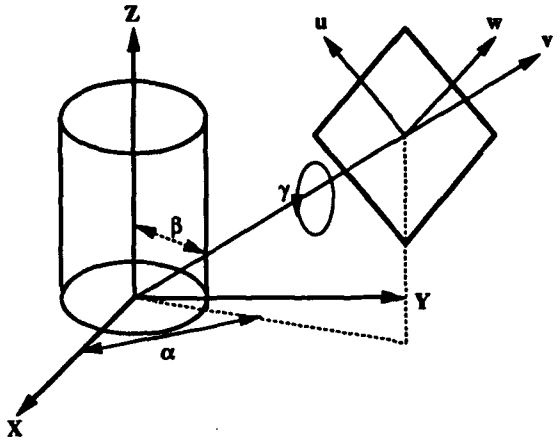


Figure 18: A coordinate system for RSHGC.

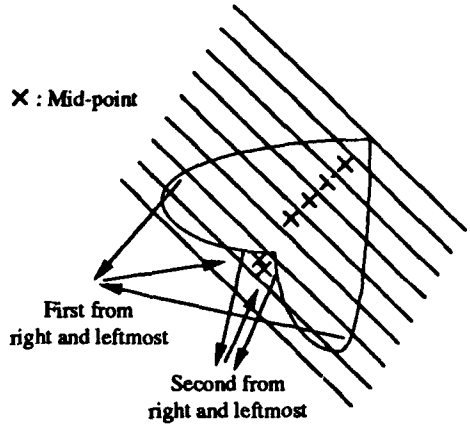


Figure 19: Mid-points and contours for skewed symmetry detection.

5.2 A simple method for finding skewed symmetry in cross-section

Researchers reported methods to find the axes of skewed symmetry, e.g. a method based on moments [Friedberg 86], or one based on local signature [Ponce 90]. In this section, we propose an alternate algorithm based on least squares fitting of a straight line to the axis points. Though it is applicable only to a single closed curve like object boundary, the criterion, linearity of the axis, is straight forward and easy to understand. Also, it doesn't depend on the curvature measurement, so that it is applicable even if the measurement at edgels is poor. The algorithm shown below is a variation of Nevatia and Binford's axis finding algorithm and is summarized as follows:

1. Discretize orientation for projection direction,
2. *For each direction*,
   (a) project edgels into buckets,
   (b) order edgels in each bucket,
   (c) if the number of edge segments in each bucket is even, calculate mid-points as shown in figure 19,
   (d) fit a straight line to the mid-points by least squares method and get the averaged error from the line.
3. find local minimum of the errors in all the projection directions; if the error value at the local minimum is sufficiently small, the fit line gives the axis of symmetry and the projection angle determines the direction of the axis of skew.

Examples of the axes of symmetry detected by the algorithm are shown in figure 17.

5.3 Experimental results of the recovery of 3D shape

By using equations in section 5.1, the original cross-section shapes and sweeping rules are calculated from the axes and edges shown in figure 15 and figure 17. The results are shown in figure 20. As in pivot theorem [Shafer and Kanade 83], it is impossible to determine an axis from meridians when the sweeping rule is a linear function. In figure 15 (b), it is assumed that the axis of SHGC passes through center of the cross-section. The recovered shape functions are good approximations of the original cross-sections and sweeping rules, however, the effects due to the deformation by perspective projection are not negligible. For quantitative analysis of original shapes, consideration of perspective projection may be required. However, for selecting hypotheses in model-based vision systems, the resulting information may be sufficiently accurate; many features used to index object shapes, such as elongatedness, compactness, area, or moments can be estimated from it.

6 Conclusion

In this paper, we presented a set of modules to extract partial descriptions of SHGC objects in an edge image and described the algorithms used in the modules, i.e., an algorithm to find the ends of SHGC, one to find the meridian edges, one to find the cross-section edges, and one to recover the sweeping rule and the original cross-section shape.

Major contribution of this work is that it shows the feasibility to extract geometrical edges from real images and provides the algorithms for SHGC objects. We believe that this work will accelerate the development of model-based vision systems and other methodologies in this field, especially their application to real images.

BUILDER-I's weak points are as follows:

Problem (i) it requires heavy computation to find geometrical edges, especially end edges;

Problem (ii) it cannot handle considerable obscurations in geometrical edges;

Problem (iii) some of the modules lack generality, in other words, requires additional restrictions to object shapes.

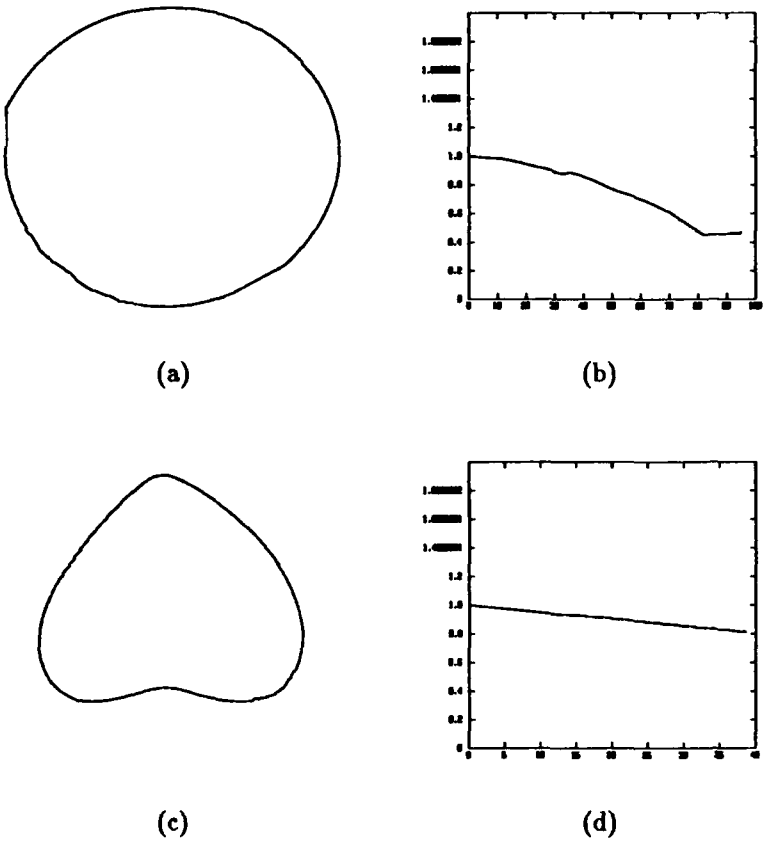


Figure 20: Recovered cross-sections and sweeping rule functions. (a): cross-section for RiceBowl, (b): sweeping rule for RiceBowl, (c): cross-section for HeartCup, (d): sweeping rule for HeartCup.

Curve fitting to the edges or selection of edgels which represents edges, e.g., B-spline representation in [Saint-Marc and Medioni 90] or zero curvature points in [Ponce et al. 89], should be effective to improve the problem (i). Also, detection of curvilinearity or super-edges which are the groups of edges that preserve C^1 or C^2 continuity, will solve the problem (i) and (ii) partly. Improvement of junction finder algorithm will not only reduce the computation in meridian finder module but also give additional clues regarding the junctions. They are especially useful for the occlusion problem and cross-section finder if it can determine the types of junctions in [Malik 87].

Though the assumptions imposed on the object shapes in 3D-recoverer module are not realistic in many situations, it is still applicable to many industrial objects and gives an example of further bottom up processings. Other researchers showed promising results for the reinforcement in the 3D recoverer to relax the restrictions on object shapes [Ulupinar and Nevatia 90] [Gross and Boult 90].

A complete system for image recognition is supposed to have a flexible control mechanism which combines bottom up and top down processings with focus of attention. So, the bottom up processing module itself should be extended to cope with various top down information, i.e., restriction on the shape or the appearance of objects in the image.

A Geometrical constraints on the meridian edges

According to [Ponce et al. 89], the projected contours of an SHGC including oblique case are:

$$\overrightarrow{OQ} = \rho(\theta) r(z) \vec{d} + z\vec{b},$$

$$\begin{aligned} \vec{b} &= \sin\beta_0 \sin(\alpha_0 - \alpha)\vec{w} \\ &\quad + [\cos\beta_0 \sin\beta - \cos\beta \sin\beta_0 \cos(\alpha_0 - \alpha)]\vec{u}, \\ \vec{d} &= \sin(\theta - \alpha)\vec{w} - \cos\beta \cos(\theta - \alpha)\vec{u}, \end{aligned}$$

where $\overrightarrow{OQ}$ is the position in an image, $(\vec{w}, \vec{u})$ is unit bases of the imaging plane, $\rho(\theta)$ is a cross-section curve, $r(z)$ is a sweeping rule, (α, β) is viewing angles, and (α_0, β_0) is orientation of the axis of SHGC.

The vector between a co-parallel edgel pair $\overrightarrow{Q_1Q_2}$ is:

$$\overrightarrow{Q_1Q_2} = [\rho(\theta_2)\vec{d}(\theta_2) - \rho(\theta_1)\vec{d}(\theta_1)] r(z).$$

So, they are parallel for a pair of meridians regardless of z. The vector between a co-meridian edgel pair $\overrightarrow{Q_0Q_1}$ and its extension $\overrightarrow{OQ_0} + t\overrightarrow{Q_0Q_1}$ are:

$$\overrightarrow{Q_0Q_1} = \rho(\theta)\vec{d}(\theta)(r(z_1) - r(z_0)) + (z_1 - z_0)\vec{b}.$$

$$\begin{aligned}\overrightarrow{OQ_0} + t\overrightarrow{Q_0Q_1} &= \rho(\theta)\vec{d}(\theta)[r(z_0) + t(r(z_1) - r(z_0))] \\ &\quad + t(z_1 - z_0)\vec{b}.\end{aligned}$$

Thus, we get the position of focus $\overrightarrow{OF}$:

$$\overrightarrow{OF} = \frac{r(z_0)}{r(z_0) - r(z_1)}(z_1 - z_0)\vec{b}.$$

where $\vec{b}$ is the projection of the 3D axis of the SHGC. Thus, the foci, i.e.,extension intersections are in the axis.

References

[Agin and Binford 73] Agin, G.J., and Binford, T.O., "Computer description of curved objects," Proc. Third IJCAI, pp.629-635, Aug. 1973.

[Binford 71] Binford, T.O., "Visual perception by computer", Proc. IEEE Conf. on Systems and Control, Miami, December, 1971.

[Binford et al. 87] Binford, T.O., Levitt, T., and Mann, W., "Bayesian inference in model-based machine vision," Proc. Workshop on Uncertainty in Artificial Intelligence, 1987, pp.73-95.

[Brady and Asada 84] Brady, J.M. and Asada, H., "Smooth local symmetries and their implementation," International Journal of Robotics Research, vol.3, no.3, 1984, pp.36-60.

[Brooks 81] Brooks, R.A., "Symbolic reasoning among 3D models and 2D images," Artificial Intelligence 17, pp. 285-348, 1981.

[Friedberg 86] Friedberg, S.A., "Finding axes of skewed symmetry," Journal of Computer Vision, Graphics, and Image Processing, 34(2), pp.138-155, 1986.

[Gross and Boult 90] Gross, A. D. and Boult, T. E., "Recovery of generalized cylinders from a single intensity view," Proc. of the Image Understanding Workshop, 1990, pp.557-564.

[Hough 62] Hough, P.V.C., "Method and means for recognizing complex patterns," U.S. Patent 3,069,654, 1962.

[Malik 87] Malik, J., "Interpreting line drawings of curved objects," Journal of Computer Vision, vol.1, no.1, 1987, pp.73-103.

[Mohan and Nevatia 89] Mohan, R. and Nevatia, R., "Perceptual organization for segmentation and description," Proc. of the Image Understanding Workshop, 1989, pp.415-424.

[Nevatia and Binford 77] Nevatia, R. and Binford, T.O., "Description and recognition of complex curved objects," Artificial Intelligence 8, pp. 77-98, 1977.

[Ponce 90] Ponce, J., "On characterizing ribbons and finding skewed symmetries," Computer Vision, Graphics, and Image Processing, vol.52, no.3, pp.328-340, 1990.

[Ponce et al. 89] Ponce, J., Chelberg, D., and Mann, W., "Invariant properties of straight homogeneous generalized cylinders and their contours," IEEE trans. on PAMI, vol.11, no.9, 1989, pp.951-966.

[Rao and Medioni 88] Rao, K. and Medioni, G., "Useful geometric properties of the generalized cone," Proc. of Computer Vision and Pattern Recognition, 1988, pp.276-281.

[Rao and Nevatia 87] Rao, K. and Nevatia, R., "From sparse 3-D data directly to volumetric shape description," Proc. of the Image Understanding Workshop, 1987, pp.360-369.

[Saint-Marc and Medioni 90] Saint-Marc, P. and Medioni, G., "B-spline coutour representation and symmetry detection," Techinical Report IRIS262, University of Southern California, 1990.

[Sato and Binford 92] Sato, H. and Binford, T.O., "On finding the ends of SHGCs in an edge image," Proc. of Image Understanding Workshop, 1992.

[Shafer and Kanade 83] Shafer., S. A. and Kanade., T., "The theory of straight homogeneous generalized cylinders," Technical Report CS-083-105, Carnegie-Mellon University, 1983.

[Sumanaweera et al. 88] Sumanaweera, T., Healey, G., Lee, B., Binford, T.O., and Ponce, J., "Image segmentation using geometric and physical constraints," Proc. of the Image Understanding Workshop, 1988, pp.1091-1099.

[Ulupinar and Nevatia 90] Ulupinar, F. and Nevatia, R., "Recovering shape from contour for SHGCs and CGCs," Proc. of the Image Understanding Workshop, 1990, pp.544-556.

An Example of 3-D Interpretation of Images using Bayesian Networks*

Wally Bishop Mann and Thomas O. Binford

Computer Science Robotics Laboratory
Stanford University
Stanford, CA. 94305

Abstract

We present preliminary results of automated image interpretation using Bayesian networks. With only a rudimentary implementation of our approach, we successfully find structure, pose and shape of a 3-D pipe elbow. Bayesian aggregation nodes are used to infer interpretations of image data from the bottom up, penultimately producing a cylinder and helix, between which simple automated symbolic algebra infers 3-D relations. In a second part, we briefly introduce Classics, a highly typed, constraint system for geometric modeling. And though not robustly developed for this early report, we mention model based prediction and how it cooperates with aggregation.

1. Introduction

This paper presents a preliminary implementation of an automated image interpretation system. For this paper we define *Interpretation* as the process of hypothesizing structure, pose and shape of three dimensional scenes, whose projection appears in a monocular image. In our framework, image interpretation has three primary components: *aggregation*, *indexing*, and *prediction*, and is guided by a *model data base* and a *control strategy* (see Figure 1). We will refer not only to interpretations of 3-D scenes, but also to interpretations of 2-D data as well.

The first part of this paper focuses on aggregation as diagramed in boldface in Figure 1. Aggregation is the grouping of structure at one level to form higher level structure, creating more complex interpretations of the bottom level data as it goes (for example, grouping lines into parallel lines or grouping parts of an object to form the whole).

Successful aggregation virtually solves the matching problem by eliminating it. Beginning with the image data itself, for in a broad sense aggregation includes edge detection and segmentation, low level interpretations rise into 3-D interpretations with model/data correspondences intact. However, two facts daunt bottom up aggregation: there exists no unique interpretation of image data, and errors, from occlusion, noise, shadows and specularities, give rise to missing or misleading evidence.

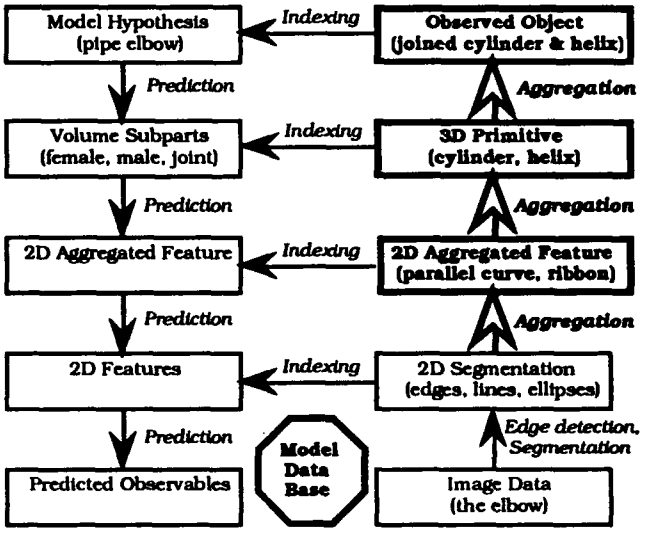


Figure 1: Diagram of Successor vision system. This paper will focus on aggregation and the modeling system, shown in boldface.

To overcome multiple image interpretations, we invoke strong generic arguments to hypothesize likely interpretations. For example, while an ellipse is not guaranteed to be the projection of a circle, it's a good bet and a worthy guess. Naturally, when combined with error inherent in the data, this gives rise to incorrect hypotheses. But the belief network dynamically accumulates evidence, pro and con, and keeps interpretations at all levels rank ordered by posterior probability. Our aim is to demonstrate that the rich information in images ultimately leads to a correct interpretation.

Interpretation is model guided at all levels, requiring models of classes of objects, geometry,

* Support for this work was provided in part by DARPA. Wally Mann was partially supported by a fellowship from the the American Electronics Association, F.M.C., Faculty Development Program.

geometric relations, and the imaging process, as well as specific object models. To achieve the requisite level of symbolic representation for geometric reasoning and machine interpretation, we developed Classics. Structures within Classics are highly typed, retaining significant symbolic information. By explicitly defining classes in terms of other classes using mathematical constraints, Classics works like a constraint network. The class typing allows us to reason in a limited (geometric) domain and keep closed the Pandora's box of solving the general constraint problem.

Previous Work

We are developing an experimental system, Successor, to support research in model based vision, and this work is part of that system.

The concepts in this paper follow directly from work in Bayesian reasoning for vision [Levitt et al. 89; Binford et al. 87; Agosta 88]. Aggregation nodes come from new work presented in [Agosta 91], and network building control strategies are discussed in [Levitt et al. 90]. In particular, Levitt, Agosta and Binford developed a similar framework for model-based vision using Bayesian inference in [Levitt et al. 89], which contains a more encompassing description than is found in this paper.

The approach and algorithms of this paper depend on other segmentation and aggregation results, some of which overlap this work in parallel development. Sato and Binford present aggregation results spanning from edge data to generalized cylinder hypotheses [Sato & Binford 91a; [Sato & Binford 91b].

Other researchers hypothesize generalized cylinders from edge contours [Ulupinar & Nevatia 91; [Gross & Boult 90], although they do not segment image data automatically. Although not probabilistic, feature grouping (aggregation) has been used in [Mohan & Nevatia 89; Chung & Nevatia 91; [Huttenlocher & Wayner 91]. Bayes networks have been used for vision modeling in [Jensen et al. 90]. We used a modified Canny edge detector developed by Binford and Wang [Binford & Wang].

Interesting and original work in algebraic vision used symbolic methods for "explicitly relating the shape of image contours to models of curved three-dimensional objects" [Ponce & Kriegman 89a; [Ponce & Kriegman 89b]. In [Lowe 91], Lowe uses numerical minimization to fit arbitrary curved surfaces to edge data. Both approaches solve not only for the projection transformation, but also for model parameters as well. In the former work, segmentation and matching is done by hand, whereas our work solves the segmentation and matching automatically. In a step precisely analogous with our prediction, Lowe relies on choosing edges nearby a projection of the model. However, his model is a priori close to correctly positioned.

Object modeling systems using constraints appeared in [Nguyena et al. 91], [Marefat & Kashyap 91] and [Walker et al. 87], focusing on polyhedral objects, and in [Kriegman 88]. In work similar to Classics, Yokoyama defines FREEDOM, an "object-oriented and constraint-based knowledge (system) for design object modeling", [Yokoyama].

2. Bayesian Networks

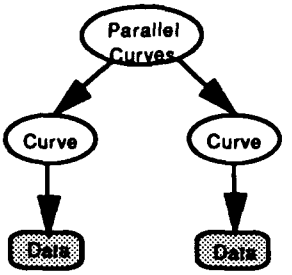


Figure 2: A simple Bayes network. (This is what the network looks like during prediction, discussed in a later section).

A Bayesian network is a graph of nodes, where nodes are random variables and arcs are conditional probabilities between the random variables. The network forms a directed acyclic graph (DAG), and we will use the term DAG interchangeably with Bayesian network. The DAG's are drawn as influence diagrams [Shacter 86], and though influence diagrams use circles to represent probabilistic nodes, we choose a slightly non-standard notation of rounded boxes for Data nodes to emphasize that they will be instantiated with a pre-posterior.

Prior probabilities, the beliefs in the states prior to additional evidence, must be explicitly provided for top level nodes (Parallel Curves in Figure 2). Priors on interior and leaf nodes arise through propagation of priors from above. Though having to provide prior probabilities seems ad hoc on first sight, if no information is otherwise available, making all prior probabilities equal in a node neutralizes their influence.

Referring to Figure 2, for example, between nodes Curve and Data is the arrow representing the conditional probability of the data given the Curve hypothesis, $P(\text{Data} = d_i/\text{Curve} = c_j)$, or the probability that the random variable Data is in state d_i given that random variable Curve is in state c_j. For notational brevity, we will usually refer to this as $P(d_i/c_j)$. In this paper, network

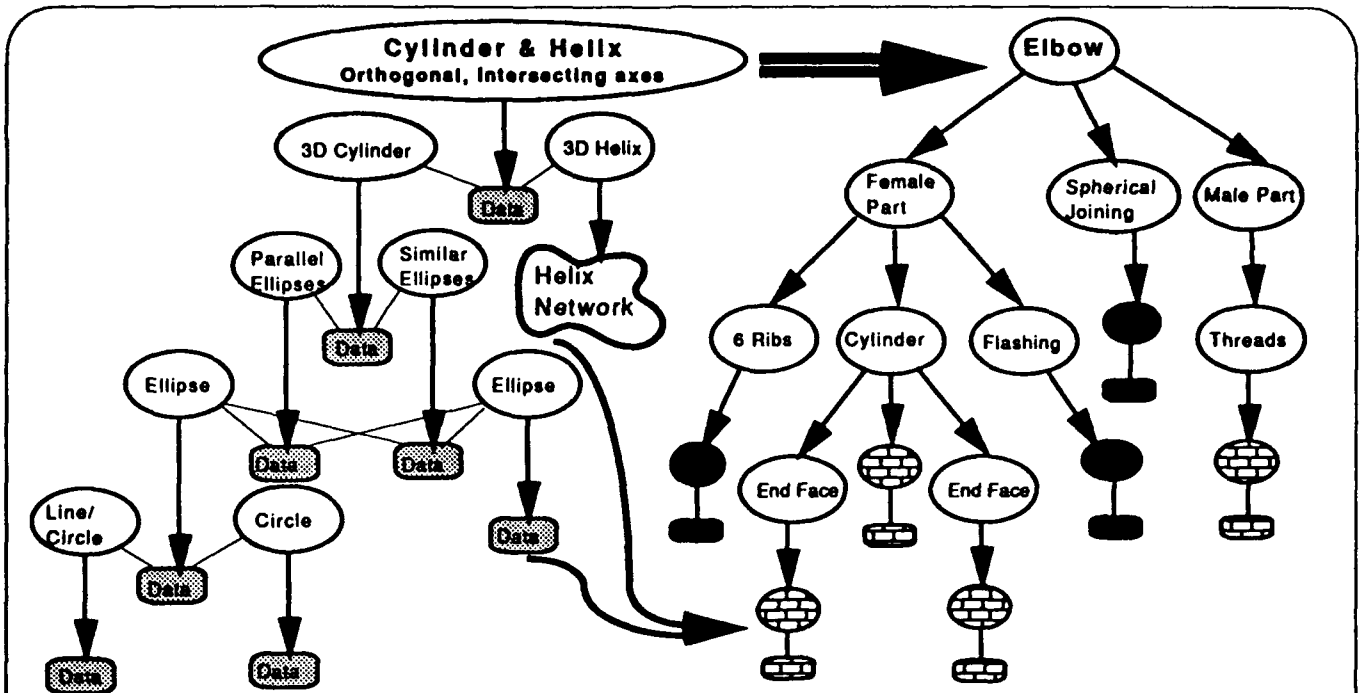


Figure 3: *An overall view of the interpretation network.* The left side shows a simplified diagram of the Bayes net used for aggregation of a 3D cylinder and helix. The right side shows prediction from the elbow model. Tiny nets from aggregation become the "ornaments" in prediction, as shown in brick. Black nodes were too weak to find in aggregation, but can be used for support and denial in prediction.

leaf nodes are always Data, whose states are known through observation, and thus we instantiate the *pre-posterior*, P(Data = Observation) = 1. Through application of Bayes Rule, the *posterior* probabilities of the remaining nodes follow and therein lies the power of Bayes networks to discriminate. We use the HUGIN Bayesian belief system to propagate probabilities through the net [Andersen, et. al. 90]. For more on influence diagrams, the reader is referred to [Shacter 86].

3. Aggregation

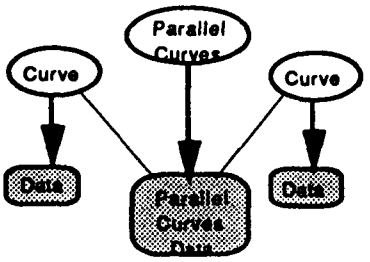


Figure 4: *An Aggregation Node.* The Parallel Curves DAG is built in response to good curve hypotheses, but is not connected to Curve nodes through conditional probabilities.

In recent work, Agosta & Levitt introduce *aggregation nodes* for combining the influence of evidence in a Bayesian network. In this paper, each level of aggregation builds a set of simple, two node DAG's (see Figure 4): a hypothesis and instantiated data node. Connected directly to the data, these small DAG's become the ornaments that we will later hang on the predictive tree [Agosta 91]. Shown in Figure 3 is an aggregation and prediction network for interpretation of the pipe elbow pictured in Figure 5.

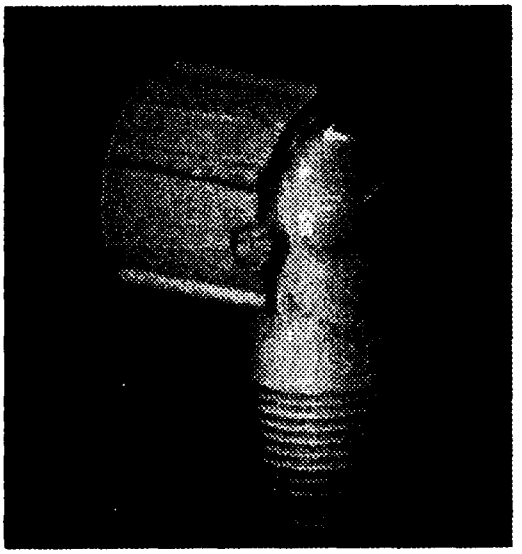


Figure 5: Image of the Elbow

3.1 Network Nodes

In our implementation, we define the following nodes (random variables): Data, Curve, Parallel-curves, Similar-curves, Parallel-&-similar-curves, Cylinder, and Cylinder-&-Helix. The probability nodes and their states are listed below along with more precise state definitions, for nodes must be well defined in order to define meaningful conditional probabilities between the nodes.

- Data nodes have states Observed and Not-observed. The Observed hypothesis is that the given feature would be observed in edge data with as least as much error as the given data.
- Curve nodes have states Line, Ellipse & Circle. The Line (Ellipse, Circle) hypothesis is that there is a linear (elliptical, circular) feature in the image data with the parameters of our Line (Ellipse, Circle) hypothesis.
- Parallel Curves nodes have states Parallel-lines, Parallel-ellipses, Parallel-circles and Not-parallel. The Parallel-lines (circles, ellipses) hypothesis is that there are two parallel linear features in the in the image data with the parameters of the hypothesis. Ellipses and circles are actually elliptical and circular arcs which are locally parallel in the manner of [Sato & Binford 91a].
- Similar Curves nodes have states Similar-lines, Similar-ellipses, Similar-circles and Not-similar. The Similar-lines (ellipses, circles) hypothesis is that there are two linear features in the image data of the same length.
- Parallel & Similar Curves nodes have states P&S-lines, P&S-ellipses, P&S-circles, and Not-P&S. The hypotheses are the conjunction of being both parallel and similar.
- Cylinder node has states Cylinder and Not-a-cylinder. The Cylinder hypothesis is that the image data is the projection of a circular cylinder with hypothesized parameters and viewing direction.
- Helix node has states Helix and Not-a-helix. The helix hypothesis is that the image data is the projection of a 3-D helix with hypothesized parameters and viewing direction.
- Cylinder & Helix has states C&H and Not-C&H. The Cylinder-&-helix hypothesis is the conjunction of cylinder and helix, with a hypothesized 3-D transformation relation between the cylinder and the helix.

Note that the Parallel-curve node has a Not-parallel hypothesis, whereas the Curve node admits no possibility that the data might not be a curve. Open world hypotheses, like Not-parallel or Not-line-ellipse-nor-circle, absorbs evidence into themselves in the simple aggregation scheme outlined in this paper. Since Parallel-lines necessarily requires two lines, for example, Not-parallel-lines will be true if *either* line is not a line, with higher levels having ever smaller posteriors on the positive hypotheses. Omitting the Not-hypothesis normalizes probabilities, and is equivalent to conditioning the node on the *existence* of (one of) the feature(s) being true. A more sophisticated aggregation network could make the existence conditioning explicit, and in particular, the prediction network should indeed express existence.

3.2 The Conditional Probabilities

Each probability node is connected to a data node with an instantiated pre-posterior, and the conditional probabilities between them are given below. Although these conditional probabilities are occasionally ad hoc, our aim is to draw upon maximum likelihood estimation, quasi-invariant analysis and imaging physics to find conditional probabilities in the lowest levels of the Bayesian network.

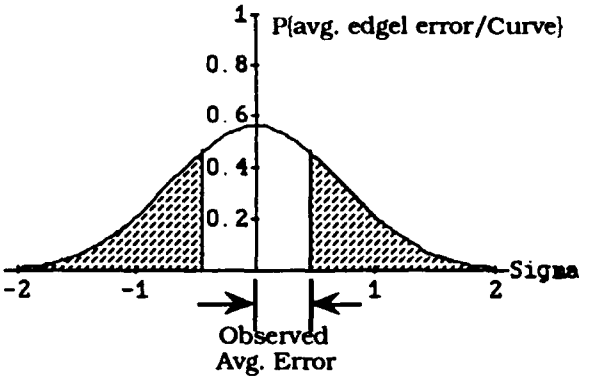


Figure 6: The shaded portion represents the probability of the Data node = Observed given a Curve hypothesis. It is the probability that the average error in the fit of the curve to edgel data is greater than the observed average error.

- P(Data/Curve): Curves are best least squares fits of the curve model to the data. Assuming the distance between each edgel and the curve hypothesis (the error of the fit) to be a Gaussian random variable, the average error of fitting a conic to edgels will also be Gaussian, shown in Figure 6. The resulting P(Observed / Curve hypothesis) is the probability that the average error of fitting a curve to edgel data is greater than the observed average error, shown shaded in the figure. The standard deviation, sigma, is estimated.

- P{Data/Parallel-curves} =
  $P\{Curve1\}*P\{Curve2\} * \left(1 - \frac{\text{angular-difference}^2}{10 \text{ degrees}}\right)$
- P{Data/Similar-curves} =
  $P\{Curve1\}*P\{Curve2\} * \left(1 - \frac{|length1 - length2|}{(length1 + length2)}\right)$
- P{Data/Parallel-&-similar-curves} = P{Parallel-curve} * P{Similar-curve}.
- P{Data/Cylinder} = P{Parallel-&-similar-ellipses}*[1 or 0], depending on whether or not the ellipses could be the projection of the ends of a circular cylinder.
- P{Data/Helix } = 1. We fit a helix to the data, but it is not in the Bayes network.
- P{Data/Cylinder-&-Helix } = P{Data/Cylinder}.

The other conditional, P{Not-observed/Feature} is naively taken to be 1 - P{Observed/Feature}. Although this is consistent with the framing of the Observed hypothesis, the choice of this conditional significantly affects the discriminatory power of the system. The choice of P{Not-observed/Feature} warrants further investigation. Bayes nets hinge on defining meaningful conditional probabilities for the network, for even when a net has intuitively pleasing structure, the conditional probabilities can be difficult to write down, if not meaningless.

3.3 Results

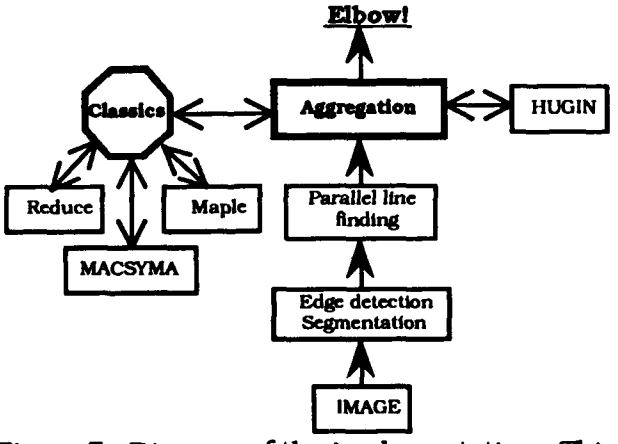


Figure 7: Diagram of the implementation. This corresponds to the right hand side of Figure 1, but with implementation details fleshed out.

The edge image from a modified version of the Canny edge detector appears in Figure 8. Rough collections of parallel curves were found by algorithms described in [Sato & Binford 91a] (Figure 7). For this subset of edges, curve hypotheses were made, as shown in Figure 9 and Figure 10.

Figure 11 shows curve hypotheses which arose by combining curves into aggregated curves, including the ellipse hypothesis which is parallel to the other end of the cylinder. As seen in Figure 10, the components of this ellipse were originally a circle and two small line segments, but simple prediction from the circle lead to the accumulation of the lines into an ellipse.

The two parallel ellipses are used to generate a 3-D cylindrical cone hypothesis and the large set of ellipses leads to a helix hypothesis. In this preliminary work, these penultimate hypotheses were a foregone conclusion and not driven by models. Also in this work, the fit of the cylinder to the data was represented by fitting two ellipses to the data, constrained to be the same shape and tilted at the same angle. The helix fit was estimated directly from the fit of the many ellipses. We anticipate fitting 3-D models directly to the data in future work.

The final model fit is shown in Figure 12. Notice that the limb edge of the cylinder matches the lower limb of the model cylinder, and could be found easily through prediction. Although it looks as though the upper limb has been missed, this is not the case, for what appears to be two candidates for the upper limb are in reality part of the small ribs attached to the cylinder. The cylinder limb is not visible and the model limb is correctly positioned between the two edges.

In order to flesh out the 3-D model, the system must establish the relative Z coordinates of the two 3-D circles to form a cylinder, and of the cylinder and the helix to form an elbow hypothesis. We used a computer interface to the Maple algebraic system to find the solution symbolically by hypothesizing a constraint that the two 3-D circles are coaxial, and that the axes of the cylinder and helix intersect. Although not model driven yet, this symbolic constraint based reasoning is the goal of Classics.

Finally, in Figure 13 and Figure 14 we show two other views of the model. Note there is a small error in the angle between the cylinder and the helix. The actual angle is 90 degrees.

The number of hypotheses at each level for the aggregation of the cylinder is given below:

Edges in image	119
Curve hypotheses	63
Aggregated curves	19
Parallel curve pairs	169
Similar curve pairs	271
Parallel & similar ellipses	12
Cylinders	3

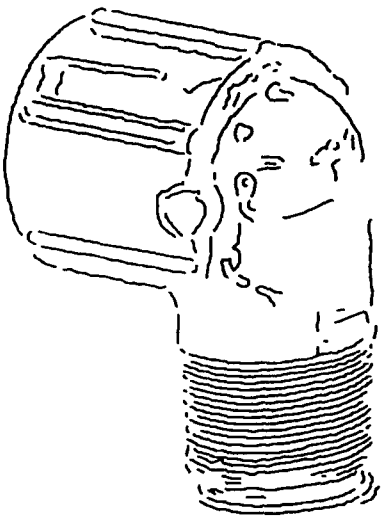


Figure 8: Edge Image

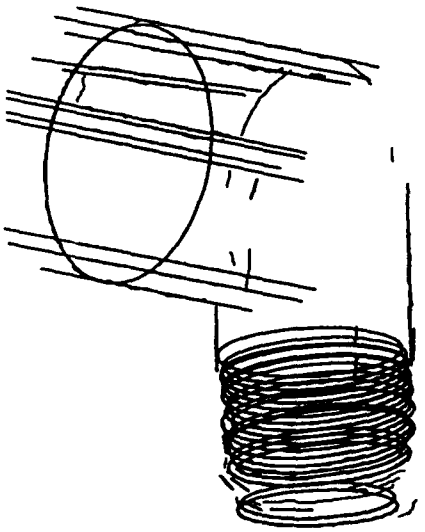


Figure 9: Initial curve hypotheses with probability greater than 0.8.

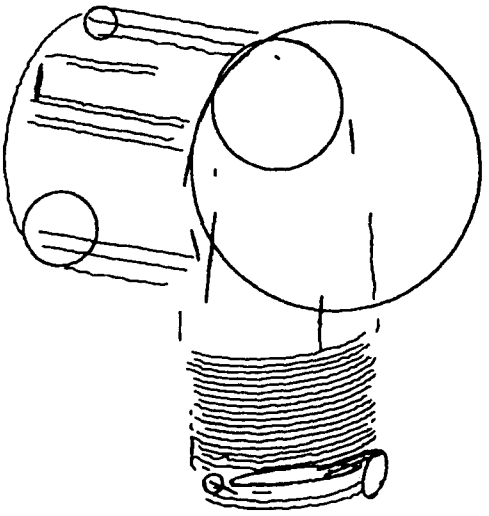


Figure 10: Initial curve hypotheses with probability less than 0.7.

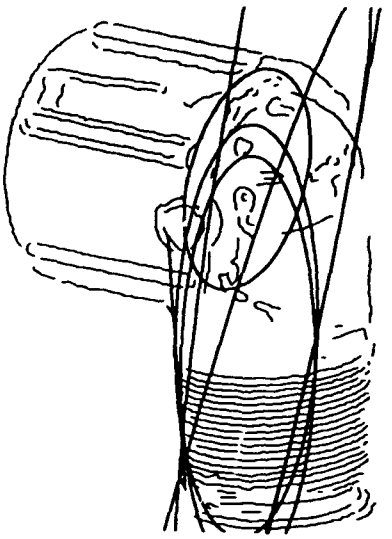


Figure 11: Aggregated curve hypotheses. These are collections of initial curve hypotheses.

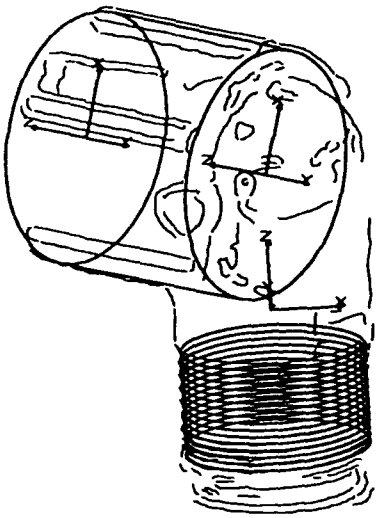


Figure 12: Final fit of a cylinder and helix to the image data.

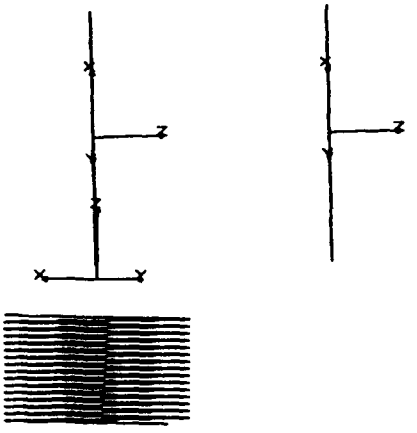


Figure 13: Model rotated for side view.

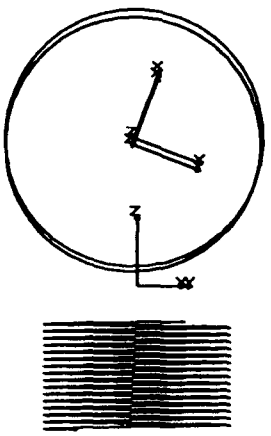


Figure 14: Model rotated for front view.

3.4 Performance

At a given level, nodes are rank ordered by posterior probability, and the most likely are considered for pairwise aggregation. Most of those considered fail a crude measure of compatibility, which is conservative in not throwing away possibly good pairs. After aggregation, even more end up with low posteriors and are never considered again. This simple scheme is order N^2, but only in the size of the subset considered for aggregation, which, if there is good data, is small. If more information is needed from the lower levels, it can be incorporated incrementally.

4. Indexing and Prediction

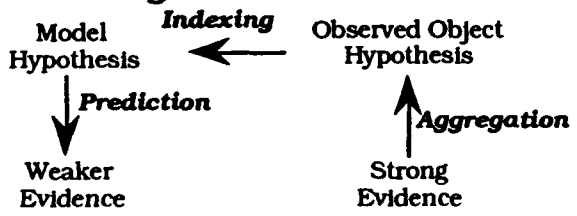


Figure 15: The fundamental cycle of image understanding presented in this work.

Implementations of indexing and prediction are limited in this work, but mentioned because, in complex images, aggregation will succeed only in partnership with indexing and prediction. In a probabilistic hypothesize and test paradigm, our method invokes models at every level to predict interpretations at the next lower level. Predictions help incorporate evidence which was not strong enough to base an aggregated hypothesis on, but which nonetheless lends evidence toward (or against) a hypothesis. This cycle, shown in Figure 15, occurs between all levels pictured in Figure 1.

The top level node in the prediction network is a model, or small set of models, from which we will derive the prediction DAG. There will be leaf nodes in the prediction DAG which have no aggregated data nodes connected to them. The viewing direction will be highly constrained because a match of 3-D data to image data has already occurred. We will project unmatched features into the image to find potential data to match and, since we know where the projection will be, there will be few possible matches and combinatorics will not be a problem.

5. The Classics Constraint System

Classics is a constraint system designed for geometric modeling. On the one hand it shares many of the features of an object oriented system: class hierarchies, inheritance of properties and maps, which resemble methods. On the other, Classics is a constraint system: classes are defined in terms of others using constraints. As much as possible, classes in Classics retain their mathematical definition. In this final regard, Classics is a symbolic mathematical language.

Classics has been designed for building a geometric modeling system. Below is a discussion of the four principle features of Classics which facilitate geometric modeling:

- Constraints
- Highly typed representation
- Symbolic mathematics
- Intuitive expression

5.1 Constraints

One defines a class in terms of other classes using constraints: Set operations (union, intersection, difference), dimension altering constraints (e.g., cartesian-product, projection), geometric constraints (e.g., parallel, coaxial, adjacent), arithmetical operations and special constraints (e.g., subclass-of, representation-of).

The constraints are themselves classes which have been defined, and have meaning to the computer. Thus it is possible for algorithms to reason about classes and the relationships between them. Classes in typical object oriented languages are data structures with simple inheritance rules, but otherwise have meaning only to the programmer.

5.2 Highly Typed Representation

Essentially "highly typed" means that everything in classics is a class. We give below two simple examples where, in the first case, data representation is unified and, in the second, declarative statements are made more natural.

Consider the following class:

Tall-People
(subclass-of People)
(> height 6')

Not only do all tall people have a height greater than six feet, the height of tall people *equals* a subclass of the class **Interval**, with a minimum of zero and maximum of six. One might think of an instance of **Tall-People**, say, **John** with height 6' 3", as a highly constrained subclass with only one member. There reason we prefer this interpretation is that it eliminates ambiguity when properties of an 'instance' have a constrained value rather than a uniquely specified value, in which an instance of **People** with height constrained to be greater than 6 would appear as something different than the class **Tall-People**.

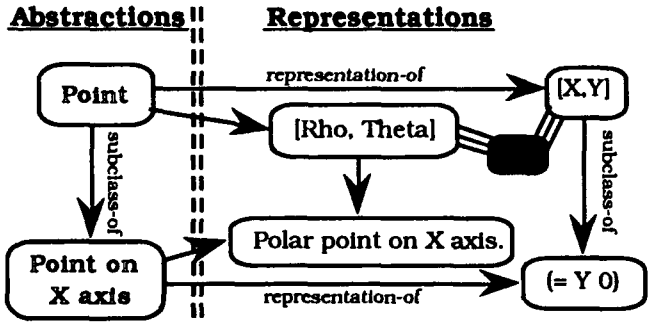


Figure 16: Simple example of the use of representations of abstract classes. The blackened class represents the map between polar and cartesian space. The system can automatically derive the class "Polar point on X axis" if needed.

Abstract classes are distinct from their representations, with isomorphic or homomorphic relations between the representations. Constraints may be given in terms of the abstract class or any representation. In the second example (Figure 16), conversions between polar and cartesian points occur automatically and the constraint (= Y 0) in the cartesian representation is equivalent to (= Theta 0) in the polar form, and the more natural representation can be used.

Some constraints can be expressed in terms independent of representation (e.g., parallel lines in terms of tangents), abstracting away details when they would be a hindrance. The abstract class collects all representations in one semantic group.

5.3 Symbolic Mathematics

Classics is built on top of several symbolic algebra packages (Maple, MACSYMA, Reduce, and later, Mathematica). Although few features are available as yet, the full functionality of these systems is potentially within classics through direct LISP calls. Translations to and from Classics expressions, which when possible match those of LISP, into the dialect for a given algebra system occur automatically.

It is not reasonable to throw general algebraic constraints at a system like Maple and expect a solution. Symbolic algebra hasn't reached that point. However, through the highly typed (object oriented) structure, Classics emphasizes *expressiveness* over automation. Classics works as a declarative programming language, letting the user provide sufficient information into the model so that Classics can solve non-trivial geometric problems without trying to be a general constraint solving system.

Classics can perform probabilistic inference with the HUGIN belief network code [Andersen, et. al. 90], and can invoke numerical fitting and minimization algorithm as well.

5.4 Intuitive Expression

Both for rigor and intuitive clarity, classes and maps have mathematically meaningful names. For example, the Common Lisp Object System (CLOS) manual gives an example of the class APPLE-PIE as having superclasses APPLES and CINNAMON[Bobrow et al. 88]. While this provides the intended effect of multiple slot inheritance, from a set theoretic perspective it makes no sense. A set with multiple superclasses is in the intersection of those superclasses. The intersection of APPLES and CINNAMON is null, not APPLE-PIE. A more precise statement might be that APPLE-PIE is the (cartesian-product-of APPLES CINNAMON), although this is clearly an inadequate definition.

6. Conclusion

We presented results from the first (almost) fully working version of the Successor image interpretation system. Beginning with grouped edge data, our contribution automatically constructed a 3-D interpretation of an image from the bottom up, using a Bayesian approach that we believe readily extends to more complex scenes. We presented details of a newly developed constraint based system for geometric modeling, which we aim to make a meta-language for guiding our interpretation algorithm without user intervention beyond model building.

Acknowledgments

My thanks to Johnmark Agosta for fruitful discussions on everything probabilistic, and to Leonid Frants for many hours of contributions to Classics. Hiroaki Sato helped me use his code

and finally, Thilaka Sumane provided useful suggestions for numerical fitting.

References

Agosta 88 J.M. Agosta, "The structure of Bayes networks for visual recognition", *Proc. 4th conf. on uncertainty in A.I.*, 1988, pp. 1-8.

Agosta 91 J.M. Agosta, "An example of a Bayes' network of relations among visual features", *Proc. International Society for Optical Engineering* (SPIE), 1991.

Andersen, et. al. 90 S. K. Andersen, et. al., "HUGIN, a shell for building Bayesian belief universes for expert systems", *Proc. 10th Intl. Joint Conf. on AI*, 1989.

Binford & Wang As yet unpublished analysis of the bias and statistics of a modified Canny edge operator.

Binford et al. 87 T.S. Levitt, T.O. Binford and W.B. Mann, "Bayesian Inference in Model-Based Machine Vision", *Proc. of the Computer Vision and Pattern Recognition Conference*, Ann Arbor, Michigan, June 1988.

Bobrow et al. 88 D. Bobrow, L. DeMichiel, R. Gabriel, S. Keene, G. Kiczales and D. Moon, "Common Lisp Object System Specification: Programmer Interface Concepts", X3J13 Document 88-002R, June, 1988.

Chung & Nevatia 91 Ronald C.K. CHung and Ramakant Nevatia, "Use of monocular groupings and occlusion analysis in a hierarchical stereo system", *IEEE Conf. on Comp. Vision and Patt. Recog.* 1991, pp. 50-56.

Gross & Boult 90 Ari D. Gross and Terrance E. Boult, "Recovery of Generalized Cylinders from a Single Intensity View", DARPA *Proc. of the Image Understanding Workshop*, 1990.

Huttenlocher & Wayner 91 Daniel P. Huttenlocher and Peter C. Wayner, "Finding Convex Edge Groupings in an Image", *IEEE Conf. on Comp. Vision and Patt. Recog.*, 1991, pp. 406-412.

Jensen et al. 90 Finn Verner Jensen, Jan Nielsen and Henrik I. Christensen, "Use of Causal Probabilistic Networks as high level models in computer vision", (Denmark: Aalborg University, Report 90-39, 1990).

Kriegman 88 David J. Kriegman, "Generic Models for Robot Navigation", *DARPA Image Understanding Workshop*, 1988.

Levitt et al. 89 T.S. Levitt, J.M. Agosta & T.O. Binford, "Model-Based Influence Diagrams for Machine Vision", *Proceedings of the 5th Conference on Uncertainty in AI*, 1989, pp. 233-244

Levitt et al. 90 Tod S. Levitt, Thomas O. Binford, and Gill J. Ettinger, "Utility-Based Control for Computer Vision", *Uncertainty in Artificial Intelligence 4*, (North Holland), 1990, PP.407-422.

Lowe 91 David G. Lowe, "Fitting Parametrized Three-Dimensional Models to Images", *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 13, no. 5, p. 441-450

Marefat & Kashyap 91 M. Marefat and R. L. Kashyap, "Geometric Reasoning for Recognition of Three-Dimensional Object Features", *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 12, no. 10, pp. 949-965, October 1990.

Mohan & Nevatia 89 Rakesh Mohan and Ramakant Nevatia, "Using Perceptual Organization to Extract 3-D Structures", *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 11, no. 11, November 1989, pp.1121-1139.

Nguyena et al. 91 Van-Duc Nguyen, Joseph L. Mundy and Deepak Kapur, "Modeling Generic Polyhedral Objects with Constraints", *IEEE Conf. on Comp. Vision and Patt. Recog.*, 1991.

Ponce & Kriegman 89a Jean Ponce & David J. Kriegman, "An Algebraic Approach to Computer Vision", Submitted to Revue d'Intelligence Artificielle, July 1989.

Ponce & Kriegman 89b David J. Kriegman and Jean Ponce, "On recognizing and positioning curved 3D objects from image contours", IEEE Trans. on Pattern Anal. and Machine Intell. vol. 12, no. 12, p. 1127-1137, 1989.

Sato & Binford 91a Hiroaki Sato and Thomas O. Binford, "On Finding the Ends of Straight Homogeneous Generalized Cylinders", To be published in *DARPA Image Understanding Workshop*, 1992.

Sato & Binford 91b Hiroaki Sato and Thomas O. Binford, "Finding and Recovering SHGC Objects in an Edge Image", To be published in *DARPA Image Understanding Workshop*, 1992.

Shacter 86 R. D. Shachter, "Evaluating Influence Diagrams", *Operations Research*, **34**, (1986).

Ulupinar & Nevatia 91 Fatih Ulupinar and Ramakant Nevatia, "Recovering Shape from Contour for Constant Cross Section Generalized Cylinders", *IEEE Conf. on Comp. Vision and Patt. Recog.*, 1991, pp. 674-676.

Walker et al. 87 Ellen Lowenfeld Walker, Martin Herman, Takeo Kanade, "A framework for representing and reasoning about three-dimensional objects for vision", *Proc. of the AAAI Workshop on Spatial Reasoning and Multisensor Fusion*, 1987, pp. 21-33.

Yokoyama Takanori Yokoyama, "An Object-Oriented and Constraint-Based Knowledge Representation System for Design Object Modeling", Institute for New Generation Computer Technology, ICOT TM-0809, 1989.

Subsection F:

Aerial Photo Matching and Interpretation

Utilization of Multispectral Imagery for Cartographic Feature Extraction

Stephen J. Ford and David M. McKeown, Jr.

Digital Mapping Laboratory
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

In this paper we discuss some potential applications for the use of multispectral imagery in conjunction with panchromatic mapping imagery. The remote sensing community has traditionally used both satellite and aerial imagery whose information content is not limited to the visible portion of the electromagnetic spectrum. Computer vision and image understanding for cartographic feature extraction has primarily been limited to panchromatic imagery. While some navigation and robotic sensing research has used sensors such as color television, acoustic and laser rangefinders, and thermal imaging, multispectral imagery has not been the subject of extensive research.

In the area of cartographic feature extraction it is clear that having surface material information could aid in monocular and stereo scene analysis. However getting such information has been difficult, particularly at spatial resolutions that are comparable with the panchromatic imagery. We describe some current work in supervised multispectral classification, the refinement of multispectral classification using monocular panchromatic imagery, and the fusion of stereo disparity maps with surface material information. Our conclusions are that multispectral imagery with moderate spatial resolution has great potential to provide scene domain cues necessary to improve the performance of cartographic feature extraction based on panchromatic imagery with high spatial resolution.[1]

1. Introduction

Over the last twenty years the remote sensing community has focused its efforts on the analysis of remotely sensed multispectral imagery such as Landsat MSS, Landsat TM, and SPOT using computational models derived from statistical analysis. As the pixel size has shrunk from 80 meters, to 30 meters, to 10-20 meters, the opportunity has increased to apply structural and spatial analysis techniques developed for high resolution aerial imagery to satellite data. Further, a new generation of high resolution airborne multispectral scanners (Daedalus, AVRIS, MEIS, etc.) can be flown to support multispectral pixel resolutions well below five meters. An opportunity exists to develop image understanding techniques to support the automated analysis of high resolution multispectral imagery that goes beyond traditional statistical analysis. In this paper we discuss some preliminary work to evaluate the utility of multispectral imagery at high spatial resolution as a source of information for automated cartographic feature extraction. The variety of applications and its potential impact for automated surface material classification, more accurate map feature attribution, and improved thematic and land-use maps, makes this an important new area of research within the context of computer vision.

This paper is organized into the following sections.[2] In Section 2 we give a brief overview of some current multispectral imaging systems, including a description of their spectral and spatial resolutions. In Section 3 we describe the current state-of-the-art in manual photointerpretation for various domain applications. These applications have been primarily in areas such as forestry, agricultural analysis, environmental studies, and regional planning. We give a brief overview of current practice in supervised classification of multispectral imagery, including issues in training set selection, band selection, and classifier types. We also

[1] This research was primarily sponsored by the U.S. Army Topographic Engineering Center under Contracts DACA72-87-C-0001 and DACA76-91-C-0014 and partially supported by the Avionics Lab, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Topographic Engineering Center, or the Defense Advanced Research Projects Agency, or of the United States Government.

[2] Figures 7 through 28 have been printed in color as an insert to be included in the 1992 DARPA Image Understanding Workshop Proceedings.

describe some tools for the generation of ground truth classifications used to assess accuracy for classifier performance evaluation. In Section 4 we describe some initial experiments in the automatic generation of surface material classification using an airborne multispectral scanner data over Washington D.C.. Finally, in Section 5 we show the application of coarse surface material classification data to improve the interpretation of stereo disparity estimates generated using high resolution panchromatic aerial imagery.

2. Multispectral Systems

The launch of the Landsat 1 satellite on 23 July 1972 marked the beginning of a new era of digital image data acquisition and processing for the remote sensing community. The principal imaging instrument onboard the first Landsat vehicle was the Multispectral Scanner (MSS) which collected image data at a spatial resolution of 79 meters simultaneously in four spectral band intervals. Table 1 lists Landsat 1 MSS spectral bandwidths. The four spectral bands measured reflected solar radiation from the earth's surface covering the electromagnetic spectrum from green to near infrared and recorded the measured energy as pixel intensity or brightness values in four separate images [Richards 86].

During the past 10 years, a trend towards higher spatial resolution for orbital multispectral imaging systems has occurred. In the early 1980's, Landsat 4 was launched carrying as part of its payload, the Thematic Mapper (TM). The Thematic Mapper, a multispectral imaging instrument, collects imagery at a higher spatial resolution of 30 meters as compared to the Landsat 1 MSS spatial resolution of 79 meters. During the latter part of the 1980's, the French satellite SPOT, carrying the High Resolution Visible Imaging Instrument (HRV), became operational and is capable of acquiring multispectral imagery at 20 meter spatial resolution. It also has a panchromatic operating mode at 10 meter spatial resolution, and the sensor can be steered to provide off axis imaging to allow for stereo coverage. In recent years, United States government organizations and professional remote sensing societies have proposed multispectral mapping satellite systems to provide high spatial resolution (10 meter) stereoscopic and multispectral coverage of the Earth's surface [Colvocoresses 90, Light 90]. The ability to collect multispectral imagery at these spatial resolutions provides the scale needed to enable the application of spatial and structural analysis to complex scenes in urban/suburban environments.

2.1. Daedalus Airborne Thematic Mapper

Until a multispectral satellite system, such as the proposed Landsat 7, becomes operational, the remote sensing community relies upon airborne systems to collect high resolution multispectral imagery. The Daedalus Airborne Thematic Mapper (ATM) is an example of an aircraft-based multispectral imaging system. The following sections cover the spectral and spatial characteristics of this multispectral imaging system along with two operational orbital multispectral satellite systems.

2.1.1. Spectral Bandwidths

The Daedalus ATM is an airborne system configured to acquire multispectral imagery in the NASA-selected Landsat TM bands and additional spectral bands not available on the Thematic Mapper [Daedalus 83]. Figure 1 illustrates the spectral bandpasses, used in collecting reflected electromagnetic radiation from the earth's surface, for the Daedalus ATM along with two popular orbiting satellite multispectral imaging systems, Landsat TM (TM) [Engel and Weinstein 83, NASA 84] and SPOT HRV [Chevrel, et. al. 81, Courtois and Traizet 86]. Tables 2, 3 and 4 give the detailed spectral bandwidths for these systems.

Figure 1 is a schematic version of the average atmospheric transmission curve from the earth's surface to the top of the atmosphere. The bandpasses of the three major multispectral imaging systems have been included to show their relative positions. The atmosphere interacts with the propagating electromagnetic radiation in two ways [Slater 80]:

- Scattering: Electromagnetic radiation is reflected or refracted by particles in the atmosphere which may range in size from gas molecules to dust particles and large water droplets.
- Absorption: Electromagnetic radiation is selectively absorbed by gas molecules contained in the atmosphere.

Atmospheric scattering and absorption processes can severely attenuate the electromagnetic radiation as it passes through the atmosphere. For example, the atmosphere heavily attenuates electromagnetic radiation in the spectral region between 1.80 - 1.95 microns due to a strong water vapor absorption band located at 1.9 microns. As a consequence, these processes dictate the electromagnetic regions where passive optical remote sensing systems can collect reflected solar radiation from the earth's surface. These electromagnetic regions are commonly referred to as atmospheric windows and have been labeled in Figure 1 as:

- Visible (VIS) (BLUE, GREEN and RED).
- Near Infrared (NIR_1, NIR_2 and NIR_3).
- Shortwave Infrared ($SWIR_1$ and $SWIR_2$).

Using these electromagnetic regions, the spectral bandpasses of the three multispectral imaging systems can be characterized as follows:

- Daedalus ATM possessing 6 visible, 2 near infrared and 2 shortwave infrared bands.
- Landsat TM collecting in 3 visible, 1 near infrared and 2 shortwave infrared bands.

Landsat 1 Multispectral Scanner (MSS)		
Band Number	Spectral Bandwidth (microns)	Electromagnetic Region
4	0.500 - 0.600	Visible (GREEN)
5	0.600 - 0.700	Visible (RED)
6	0.700 - 0.800	Near Infrared (NIR_1)
7	0.800 - 1.100	Near Infrared (NIR_1-NIR_2)
Spatial Resolution: 79 meters		

Table 1: Reflective Spectral Bandwidths for Landsat 1 MSS

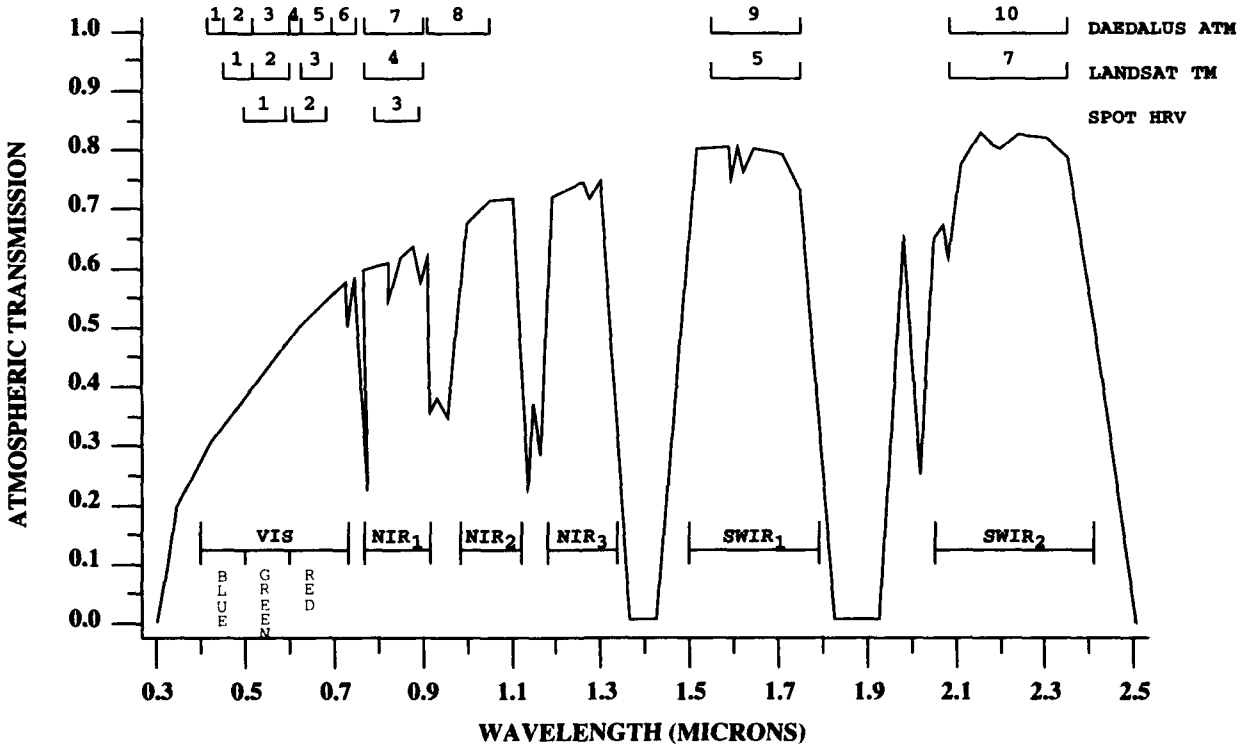


Figure 1: Reflective Spectral Bandpasses of Daedalus ATM, Landsat TM and SPOT HRV Multispectral Imaging Systems

• SPOT HRV imaging with 2 visible and 1 near infrared bands.

With the Daedalus ATM collecting reflected solar energy in 10 spectral regions, spanning the visible to shortwave infrared, its multispectral imagery provides additional measurements in the visible and near infrared electromagnetic regions.[3] These measurements are not available in Landsat TM or SPOT HRV multispectral imagery and provide additional spectral characterization of earth surface materials.

2.1.2. Spatial Resolution

Since the Daedalus ATM multispectral imaging system is aircraft-based, the sensor has the capability to acquire high resolution multispectral imagery by flying the system at lower altitudes. When comparing the Daedalus ATM collection set spatial resolution to Landsat TM or SPOT HRV multispectral imagery, the Daedalus ATM imagery (8 meter) possesses considerably higher spatial resolution than either of the

[3] It also has an eleventh channel in the thermal region between 8.5 and 13.0 microns whose response is sensitive to emitted energy from natural and man-made surfaces.

Daedalus Airborne Thematic Mapper (ATM)		
Band Number	Spectral Bandwidth (microns)	Electromagnetic Region
1	0.420 - 0.450	Visible (BLUE)
2	0.450 - 0.520	Visible (BLUE)
3	0.520 - 0.600	Visible (GREEN)
4	0.605 - 0.625	Visible (RED)
5	0.630 - 0.690	Visible (RED)
6	0.695 - 0.750	Visible (RED)
7	0.760 - 0.900	Near Infrared (NIR_1)
8	0.910 - 1.050	Near Infrared (NIR_2)
9	1.550 - 1.750	Shortwave Infrared ($SWIR_1$)
10	2.080 - 2.350	Shortwave Infrared ($SWIR_2$)
Spatial Resolution: 8 meters		

Table 2: Reflective Spectral Bandwidths for Daedalus ATM

two satellite systems (30 and 20 meter, respectively)[4] As a result of the Daedalus ATM 8 meter spatial resolution, detailed spatial scene features are present in the Daedalus image structure which are not evident in Landsat TM or SPOT HRV multispectral imagery. For example, Figures 7, 8, and 9 demonstrate this difference in spatial resolution between the Daedalus ATM imagery and *simulated* SPOT HRV and Landsat TM multispectral imagery for a near infrared image (Daedalus ATM band 7, 5, and 3). The simulated spatial resolution for the SPOT HRV and Landsat TM image were created by spatially reducing the Daedalus imagery.

In the Daedalus image (Figure 7), features such as buildings, road networks, shadows, and ground structures (e.g. reflecting pools, treetops, lawn area, parking lots, etc.) can be readily identified. However, in the simulated SPOT HRV image (Figure 8), these features are no longer easily discernible. For the simulated Landsat TM image (Figure 9), only significant areal features such as the mall area, large water bodies, and city blocks are readily identifiable.

3. Multispectral Photointerpretation

One advantage of multispectral imagery over panchromatic (i.e. black and white) imagery is the ability to view the imagery in various combinations of color presentations. For visual interpretation tasks, this flexibility enables the image analyst to select the appropriate band combinations which provide the best visual discrimination between land-cover types of interest [Hoffer 78].

The utilization of multispectral imagery to aid in identifying and mapping land-cover types crosses many disciplines. These disciplines range from forestry to regional planning and their particular interpretation tasks are just as varied. As a consequence, each discipline must determine the appropriate land-cover types in order to address their task at hand. The following brief discussion gives an overview of a variety of remote sensing applications from various disciplines.

Forestry issues include the identification of tree species, measurement of the damage due to insects, pollution, environmental stress, and forestry management [Goodenough, et. al. 87, Hopkins, et. al. 88]. There has been recent interesting work [Ahern, et. al. 91] to compare the performance of conventional digital classification techniques and visual interpretation of digitally enhanced monotemporal SPOT HRV multispectral images for the identification of tree species.

Crop analysis and agricultural studies were one of the first major justifications for the Landsat 1 and 2 programs. Issues include diverse applications from vegetable crops census [Williams, et. al. 87] to use of airborne Daedalus ATM data to map the extent of irrigated crops [Williamson 89].

[4] As stated previously SPOT has a 10 meter panchromatic capability. The focus of this work is multispectral imagery with high spatial resolution.

Landsat Thematic Mapper (TM)		
Band Number	Spectral Bandwidth (microns)	Electromagnetic Region
1	0.450 - 0.520	Visible (BLUE)
2	0.520 - 0.600	Visible (GREEN)
3	0.630 - 0.690	Visible (RED)
4	0.760 - 0.900	Near Infrared (NIR_1)
5	1.550 - 1.750	Shortwave Infrared ($SWIR_1$)
7	2.080 - 2.350	Shortwave Infrared ($SWIR_2$)
Spatial Resolution: 30 meters		

Table 3: Reflective Spectral Bandwidths for Landsat TM

SPOT High Resolution Visible (HRV) Imaging Instrument		
Band Number	Spectral Bandwidth (microns)	Electromagnetic Region
1	0.500 - 0.590	Visible (GREEN)
2	0.610 - 0.680	Visible (RED)
3	0.790 - 0.890	Near Infrared (NIR_1)
Spatial Resolution: 20 meters		

Table 4: Reflective Spectral Bandwidths for SPOT HRV

Land use analysis and regional planning often integrate geographic information systems, image processing, and multispectral classification [Ehlers, et. al. 90]. Land-cover inventories [Baumann 90] have been used to monitor transitional land-use from agricultural to second-home property. Wildlife management applications include estimation of bird foraging acreage [Hodgson, et. al. 87] to habitat suitability models for deer [Ormsby and Lunetta 87].

In the remainder of this section we briefly overview statistical classification techniques and describe our development of a supervised spectral classifier.

3.1. Statistical Classification Techniques

With the collection of reflected energy in several spectral regions as illustrated in Figure 1, multispectral imaging systems provide an insight into a material's spectral signature. Using the spectral energy measurements from a set of multispectral bands, individual multispectral image pixels can be assigned or classified into spectral classes (e.g. grass, tree, water, soil, etc.) of similar measurements, based on the multispectral image pixel's intensity or brightness values [Sabins 87]. Traditional multispectral classifiers can be categorized into one of two methods: unsupervised and supervised. The primary distinction between the two multispectral classification procedures centers around the involvement and interaction of the image analyst with the classification process. Typically, time must be spent by the image analyst to identify candidate spectral classes, called training sets, prior to supervised classification. However in both classification procedures, input band selection is an issue common to both methods. In the following sections, we briefly discuss both methods and describe our development of a traditional supervised classifier along with post-classification evaluation tools.

3.2. Unsupervised Classification of Multispectral Imagery

Unsupervised classifiers assign multispectral image pixels to spectral classes without the image analyst having foreknowledge of the existence or names of those spectral classes. This classification procedure most often uses clustering methods (i.e. grouping spectrally similar pixels together) to determine the number and location of the spectral classes into which

the image data falls and to determine the spectral class of each pixel. The image analyst then identifies and labels those spectral classes *a posteriori* [Richards 86, Swain and Davis 78].

3.3. Supervised Classification of Multispectral Imagery

Supervised classification is the procedure most often used for quantitative analysis of remote sensing image data. For supervised classifiers, spectral classes are identified before multispectral classification occurs. First an image analyst or domain scientist selects a representative set of pixels, commonly called a training set, to statistically describe each of the desired spectral classes. These training sets establish potential spectral classes in multispectral feature space to which the image pixels can be assigned. During the actual classification phase, each individual image pixel is compared to each spectral class and assigned to the spectral class to which the image pixel has highest likelihood of being a member [Richards 86].

3.3.1. Training Sets

The selection of appropriate land-cover types or spectral classes by an image analyst is based upon his/her domain knowledge, the scene content, and the objectives of the classification. Once a list of thematic classes has been established, training regions are selected in the multispectral image. Traditionally, for supervised training, the training sets are contiguous pixels or blocks of pixels from representative locations across the image called block training [Richards 86]. An alternative to block training includes single-pixel training set selection [Gong and Howarth 90], where individual pixels in the training set are several pixels away from any other pixel member of the training set.

Statistical measures are calculated for each spectral class training set. These statistical measures include minimum vector, maximum vector, mean vector, covariance matrix and correlation matrix. The dimensionality of the vectors and matrices is determined by the number of bands used in the statistical analysis. Of particular interest are the number of samples, mean vector, and covariance matrix with the latter two metrics being directly used in the supervised spectral classifier. Sufficient training samples for each spectral class must be present to allow reasonable estimates of the mean vector and covariance matrix [Richards 86, Swain and Davis 78].

The mean vector characterizes the spectral class's average intensity or brightness level for each multispectral band while the covariance matrix describes the shape and orientation of the spectral class's population, assuming a multivariate normal distribution. The diagonal of the covariance matrix contains the variance or dispersion of the spectral class's brightness levels for each multispectral band. To assess the amount of correlation between any two multispectral bands for a given spectral class, the correlation matrix can be consulted. By normalizing the covariance matrix, a correlation matrix is generated with coefficients ranging from -1.0 to 1.0, inclusive. A pair of highly correlated bands will have correlation coefficients near -1.0 or 1.0 while an uncorrelated band pair will have coefficients close to 0.0. Highly correlated band pairs indicate that a pixel's brightness level in band X can be predicted, within some delta, given the pixel's brightness level in band Y. Therefore, techniques which add additional bands to improve class discrimination will fail if the bands are highly correlated with one another.

3.3.2. Band Selection

When performing spectral classification, the number of input multispectral bands along with spectral region locations of the bands will influence the classification process, in terms of run time and the ability to discriminate between the candidate spectral classes. Therefore, some care must be taken when selecting the input multispectral bands with respect to the candidate spectral classes. Common practice include the selection of one band from the visible, the near infrared and shortwave infrared. Such a set forms a good basis for discrimination between water, soil, vegetation and man-made spectral features. A fourth band from any of these three spectral regions may be included for additional discrimination between candidate spectral classes.

It is best to avoid including band pairs which are highly correlated, if possible. The correlation matrix generated from the spectral class's training set should be reviewed for obvious high correlation between band pairs. Typically, adjacent spectral bands in the electromagnetic spectrum tend to demonstrate high correlation. There are several established metrics of separability (i.e. the degree of distinguishability between spectral classes) available to predict best multispectral band combinations for spectral classifications [Mausel, et. al. 90, Richards 86, Swain and Davis 78]. These metrics are based on measurements of the statistical separability between spectral classes of interest [Kailath 67, Swain and King 73].

3.3.3. Classifiers in The Large

Spectral classifiers for the quantitative analysis of remotely sensed image data can range from complex to very simplistic. The most commonly used techniques include maximum likelihood, minimum distance, parallelepiped, and mahalanobis distance.

Maximum likelihood assumes that the spectral class probabilities are multivariate normal distributions. This is an assumption, rather than a demonstrable property of natural spectral classes [Richards 86]. The probability distribution of each individual spectral class is modeled by using its mean vector and

covariance matrix as calculated from its training set. When classifying a multispectral image pixel, the probability of the pixel belonging to each of the candidate spectral classes is determined and assigned to the spectral class with the highest probability.

When the number of training samples per candidate spectral class is limited, the estimation of the statistics of the class may be inaccurate, especially the covariance matrix. Under this situation, it is probably best to resort to a classifier utilizing only the mean positions of the candidate spectral classes like a minimum distance classifier. When classifying a multispectral image pixel, the distance from the pixel to the mean of each candidate spectral class is calculated and assigned to the spectral class with the smallest distance [Duda and Hart 73, Richards 86, Swain and Davis 78].

A parallelepiped classifier partitions multispectral feature space using *n*-dimensional parallelepipeds [Jensen 86]. Parallelepiped generation can be implemented using the mean vector and standard deviation or the minimum and maximum vector from each candidate spectral class. In both implementations, lower and upper endpoints are used to define the length of the parallelepiped's sides. Using the mean vector and standard deviation, the endpoints would be calculated by extending out +/- *n* standard deviations from the mean point for each of the bands in feature space. In the case of using the extrema vectors, the minimum and maximum for each band in feature space defines the endpoints. A multispectral pixel is assigned to a particular candidate spectral class if the pixel resides inside that spectral class's parallelepiped [Richards 86].

If the covariance matrices of all the candidate spectral classes are considered to be equal (i.e. a global covariance matrix), the maximum likelihood classifier reduces to the mahalanobis distance classifier. The mahalanobis distance is a statistical distance from a multispectral pixel to a candidate spectral class's mean point, normalized by the global covariance matrix [Duda and Hart 73, Richards 86]. Like the minimum distance classifier, the smallest distance between the pixel and a spectral class mean point determines the pixel's assignment. The normalization by the global covariance matrix retains a degree of direction sensitivity, like a maximum likelihood classifier, which is absent in a minimum distance classifier.

3.4. Supervised Spectral Classifier

Our supervised spectral classifier, MBCLASS, utilizes pre-classification spectral class statistics derived from the training sets and user-selected input multispectral image bands to assign each multispectral image pixel to one of the available candidate spectral classes from the training sets. A maximum likelihood decision rule, which assumes multivariate normal distributions for the candidate spectral classes, is used as the discriminant function for assigning multispectral image pixels to one of the available candidate spectral classes. Upon completion of classifying the multispectral image, a classmap is generated, recording each multispectral image pixel's class assignment. We show some examples of the results of this classification process in Section 4. We use these results as a basis for out experiments in multispectral fusion and refinement in Section 5.

An inherent problem with this assignment method, commonly called forced allocation, involves the assignment of outlying multispectral pixels to a candidate spectral class which are not representative of that spectral class. Such outliers may be due to sensor noise or saturation during image acquisition, heavily mixed pixels composed of two or more spectral ground features (e.g. vegetation and water) or absence of proper candidate spectral classes. To alleviate this inherent problem with forced allocation, a threshold limit can be specified in MBCLASS to minimize the assignment of outliers to the candidate spectral classes [Richards 86, Swain and Davis 78].

3.5. Assessing Classification Accuracy

With the generation of classmaps by various classification schemes, it is necessary to have some method for evaluating and comparing the accuracy of the generated classmaps. This classmap assessment process is required to aid in the assessment and improvement of classification schemes. Various techniques have been developed and implemented in the remote sensing community to determine and evaluate the accuracy of land-use/land-cover maps derived from remotely sensed data [Aronoff 82, Dicks and Lo 90, Fitzpatrick-Lins 81, Story and Congalton 86]. The basic accuracy assessment procedure involves the selection of samples from the land-use/land-cover map, verification of the samples using ground truth, and statistical evaluation of the verified samples from the land-use/land-cover map [Congalton, et. al. 83, Congalton, et. al. 84, Congalton and Mead 86].

Typically, in an operational cartographic environment, attempting to verify every sample in a land-use/land-cover map is prohibitively time-consuming and expensive. Therefore, a sample selection scheme is adopted to compile a representative sample population from which an accuracy estimate of the entire land-use/land-cover map can be determined. Common sample selection strategies applied to land-use/land-cover maps are simple random sampling without replacement, cluster sampling, stratified random sampling, systematic sampling and stratified systematic unaligned sampling [Congalton 88]. Arguments for and against each of these strategies can be made, and the decision as to which method to use is often application and data dependent. The sample procedure utilized should minimize the effects of

spatial autocorrelation as well as ensure that all classes of interest are adequately sampled [Dicks and Lo 90]. The selected map samples are then usually verified with aerial photographs and field site visits when photo verification is inadequate [Dicks and Lo 90, Fitzpatrick-Lins 81]. The verification results are then tabulated in an error or confusion matrix with an overall map accuracy percentage calculated [Story and Congalton 86].

We are beginning to develop tools to allow for accuracy assessment of the ground-truth classification. Currently we rely on ground truth generation using high resolution black and white aerial photographs. These hand segmentation tools are useful for generating ground truth where geometric relationships, such as building size, shape and boundaries, are the primary focus. With regard to spectral classification, the ground truth needs to represent the material types located in the scene. As a consequence, our previously developed manual hand segmentation tools were inadequate and inappropriate. We have begun to address this inadequacy by the development of an interactive supervised classification tool, ICLASS, to generate surface material ground truths in the form of reference classmaps. In the following section we discuss experiments in surface material classification using our supervised classification system, MBCLASS.

4. Experiments in Surface Material Classification

One area of current research is the automated construction of surface material classmaps in complex urban environments. The surface material classmaps are generated by our supervised spectral classifier, MBCLASS, using Daedalus ATM multispectral imagery of Washington, D.C.. This imagery was collected for the United States SPOT HRV Simulation Campaign [SPOT 83, SPOT 84] and should not be confused with actual SPOT HRV imagery at 20 meter spatial resolution having three spectral regions as outlined in Table 4. The following sections discuss the generation of surface material classmaps for two sites in the Washington, D.C. metropolitan area.

4.1. Urban Area Test Sites

Figures 10 (CIVIL1) and 11 (GAO1) show two of the test areas with which we have been experimenting. Both images are shown using a near infrared encoding using Daedalus bands 7, 5, and 3. The scene content is representative of a complex urban area with buildings of various shapes, sizes, and heights, street networks, and landscaped areas. Specific scene characteristics include:

- CIVIL1: The Civil Service Building is located in the center of the image. Other major buildings and landmarks relative to the Civil Service Building include: the Reflecting Pool to the south, the State Department to the west, the Interior Department immediately to the east and the General Services Administration to the northeast. A mixture of natural terrain and man-made features comprise the image.

- GAO1: The General Accounting Office Building and Judiciary Square occupy the right of center portion of the image. Other major buildings and landmarks located in the image include: the FBI and Justice Department in the southwest portion, the Labor Department in the southeast section and I-395 just east of the General Accounting Office Building. This image is dominated by man-made features and structures.

4.2. Training Sets

The objective of our classification task involves the generation of surface material classmaps at a *coarse* level for urban multispectral scenes. That is to say we are initially only interested in characterizing the primary level of land-cover detail. In our urban analysis problem, the two primary land-cover types of most interest to us are natural terrain and man-made features. In Figure 2, these two primary land-cover types are further divided into specific spectral classes based upon visual interpretation of the Daedalus ATM multispectral imagery. The inclusion of a shadow feature in the spectral class hierarchy alleviates misclassifications of shadow pixels as water spectral features due to spectral similarities between the two features.

Block training sets, consisting of homogeneous areas of pure pixels, were collected manually for the candidate spectral classes from various regions distributed throughout the entire Daedalus ATM multispectral imagery. The candidate spectral classes are listed in Table 5 with the number of training samples or pixels per spectral class. The combined total of spectral class training set samples represents 1.5% of this Daedalus ATM multispectral image.[5]

4.3. Band Selection

During the process of selecting training sets for pre-classification spectral class statistics, it was observed that Daedalus ATM bands 3, 5, 7, and 10 provided excellent color separation among the earth surface features (e.g. grass, tree, water, soil, etc.) and man-made features (e.g. road, bridge, building, parking lot, etc.). Therefore, from a subjective point of view, it was felt these four multispectral bands would provide fairly reasonable spectral class separation for the supervised spectral classifier, MBCLASS. As a validation check, the average Jeffries-Matusita Distance [Mausel, et. al. 90, Richards 86, Swain and

[5] The Daedalus ATM multispectral imagery over Washington, D.C. measured 716 rows by 3000 columns for each of the 10 reflective bands.

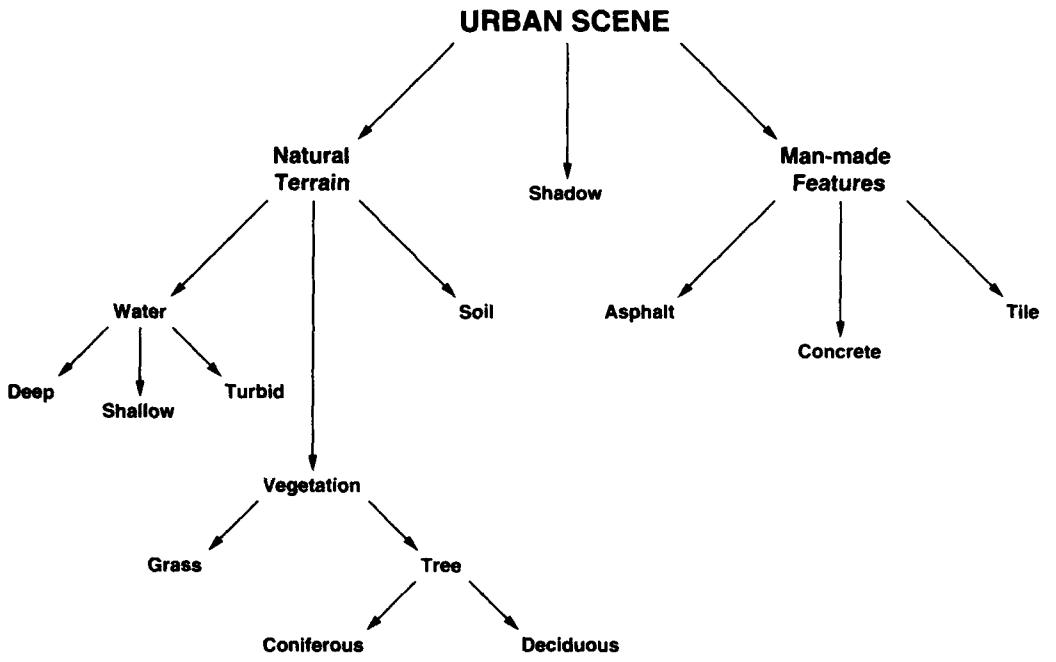


Figure 2: Urban Spectral Classes

Spectral Classes of Interest	
Spectral Class Name	Number of Training Samples
asphalt	733
concrete	656
coniferous tree	52
deciduous tree	9720
deep water	16433
grass	2535
shadow	140
shallow water	823
soil	354
tile	260
turbid water	2269
Total Training Samples: 33975	

Table 5: Spectral Classes Used in Classification

Daedalus Airborne Thematic Mapper (ATM)	
Band Combination	Average JM-Distance
3,4,7,10	1.980
2,3,4,7	1.979
3,4,6,10	1.978
3,5,7,10	1.977
3,4,7,9	1.977
3,6,7,10	1.977
3,4,6,7	1.977
2,3,4,6	1.976
3,4,5,7	1.976
2,3,5,7	1.976
Maximum Possible Distance: 2.0	

Table 6: Top 10 Four Band Combinations Using the Spectral Class Training Sets

Davis 78] was calculated, using the statistics from the spectral class training sets, in order to rank spectral class separability for all four band combinations. Table 6 lists the top 10 multispectral band combinations from this calculation for all four band combinations. The spectral class separability score for Daedalus ATM band combination 3, 5, 7, and 10 is 1.977 with a ranking of 4 out of 210 band combinations.

4.4. Generated Surface Material Classmaps

A surface material classmap was generated for each site, CIVIL1 and GAO1, using our supervised spectral classifier, MBCLASS, with forced allocation invoked. The resulting classmaps are shown in Figures 12 and

13, respectively, with the class assignment of each multispectral pixel encoded by its color. A multispectral class legend is shown in Figure 18.

The forced allocation method requires that each pixel in the multispectral image be assigned to one of the candidate spectral classes. As discussed earlier, this technique often results in the assignment of outlying multispectral pixels to a candidate spectral class which are not representative. Two examples of this occur with water and shadow pixels. Our shadow spectral class was defined by training pixels with no direct illumination and minimal downwelling illumination due to atmospheric scattering. Therefore, we would expect little surface material information to reach the sensor. Shallow water pixels along the edge of the reflecting pool in Figure 12 are classified as shadow due to the mixing of water pixels with the immediately surrounding background. The water and shadow spectral classes have similar spectral features and these mixed pixels exhibit properties closest to the shadow class. In a second case, several shadow areas along side of buildings in both test areas shown in Figures 12 and 13, are classified as shallow water. This is a result of indirect illumination (i.e. scattered solar radiation) on the asphalt street surfaces. This mixed pixel occurs at the transition edge between shadows and directly illuminated asphalt.

Traditionally, this problem is handled either via post-processing such as contextual classification, or the application of a threshold limit based on percentage confidence followed by a second stage classification [Wharton 82]. We address these problems in the following section on multispectral fusion, where we use additional information in the form of high resolution panchromatic aerial imagery to refine the classification estimates.

5. Multispectral Fusion

In this section we describe recent research to evaluate the utility of information fusion between multispectral imagery with moderate spatial resolution and high resolution panchromatic aerial imagery. We have focused on two interesting problems. The first experiment is the refinement of surface material classmaps using monocular segmentations of high resolution panchromatic imagery. The second experiment involves the fusion of high resolution stereo disparity maps with moderate resolution surface material classmaps. Our two experimental sites, CIVIL1 and GAO1, are used to illustrate the process and utility of surface material classmap refinement and fusion.

5.1. Classmap Refinement using High Resolution Panchromatic Imagery

Surface material classmaps generated by traditional spectral classifiers tend to have isolated or small groupings of pixels assigned to a class which are out of context with its immediate neighbors. For example, the CIVIL1 classmap in Figure 12 has shallow water pixels in shadow regions located on city streets and shadow pixels in the middle of waterbodies composed of shallow water pixels.

One post-classification processing technique to rectify classification inconsistencies employs a class majority operator which retains or changes the central pixel's class assignment based on the immediate neighborhood's class composition and a set of user-specified decision rules [Townsend 86]. A second technique uses context and conditional probabilities to modify classifications [Wharton 82]These post-classification processing techniques either use information encoded in the classmap to make a decision, or *a priori* knowledge concerning common mixed pixel confusions. The technique which we are pursuing utilizes ancillary spatial scene information in the form of high resolution black and white aerial imagery to perform the refinement of the surface material classmap.

5.1.1. Monocular Fusion

Figures 3 and 5 are the left images from a stereopair of high resolution black and white aerial imagery with spatial resolution of approximately 1.2 meter. These images each cover a subscene area centrally located in the multispectral images in Figures 10 and 11.

We use surface illumination information represented by the segmentation of the high resolution panchromatic images into very fine surface patches of nearly homogeneous intensity to provide a basis for classmap refinement [McKeown and Perlant 92]. Each nearly uniform intensity region identifies a surface patch having, in the ideal case, a nearly homogeneous surface material type within that region. These patches are used to guide a statistical analysis of the classmap based on the assumption that such patches correspond closely with physical surfaces of the scene. We attempt to verify or modify the class assignments encoded in the surface material classmaps by examining the classmap statistics within each of the corresponding patch regions.

5.1.2. Multisensor Registration

Before the spatial segmentation information can be applied to the classmaps, the surface material classmaps must be registered to the high resolution black and white aerial imagery. Two arising issues from the registration of the moderate resolution classmaps (8 meter) to the finer resolution aerial imagery (1.2 meter) are maintenance of geometric accuracy and preservation of the class assignment information.

The first issue is concerned with providing accurate geometric correspondence or image registration between two different image sets. The multispectral imagery was acquired by an electro-optical imaging

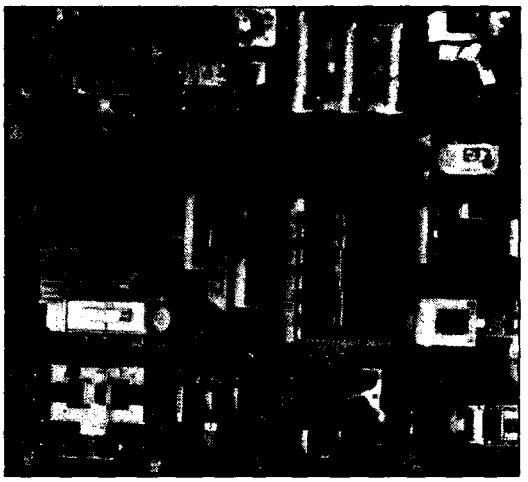


Figure 3: CIVIL1 Left Image (1.2 meter)

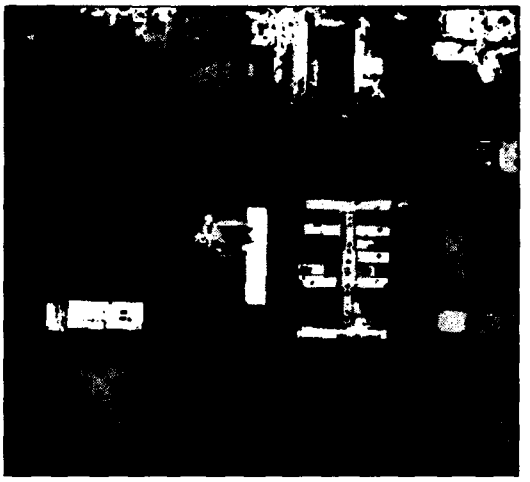


Figure 4: CIVIL1 Refined S1S2 Disparity Image



Figure 5: GAO1 Left Image (1.2 meter)



Figure 6: GAO1 Refined S1S2 Disparity Image

system while the panchromatic imagery was captured by a film-based system then subsequently digitized. Each system has its own inherent optical and imaging characteristics which affect the spatial and radiometric properties of the resulting digital image. These image properties include distortion, rotation, translation, and intensity level dynamic range. To overcome these image differences during the image-to-image registration process, we first correspond the images to a map-based reference system using a database of image chips with associated map coordinates (i.e latitude and longitude). Once this image-to-map correspondence is completed, we have the ability to determine the pixel coordinates from one image set to another image set by utilizing the image-to-map and map-to-image correspondence model in CONCEPTMAP [McKeown 84, McKeown 87, Perlant and McKeown 90]. This model provides the geometric image registration between the classmaps and panchromatic imagery.

The second issue of preserving the class assignments in the classmaps is equally as important. Resampling of the classmap to the aerial imagery must preserve the original intensity levels of the classmaps since these intensity levels encode the class assignment at each pixel location. A registration procedure utilizing interpolation would destroy the encoding scheme and result in erroneous pixel class assignments. Therefore, a nearest-neighbor assignment procedure is used to ensure the preservation of the classmap's original class assignment at each pixel location.

Figures 14 and 15 show the area of image correspondence between the high resolution aerial imagery (1.2 meter) and the generated surface material classmaps (8 meter). To register the classmaps to the high resolution aerial imagery, the image-to-map and map-to-image correspondence model provided by CONCEPTMAP's landmark database is employed, resulting in the registered surface material classmaps shown in Figures 16 and 17. The registered surface material classmaps appear blocky due to the effects of using image-to-image pixel correspondence between

the 8 meter classmaps and 1.2 meter aerial imagery using nearest-neighbor resampling.

5.1.3. Refinement

Once we have registered the imagery, we can use the region boundaries in the panchromatic imagery to select classmap values from the registered classmap imagery. A histogram of classmap values within each region is computed and a statistically significant value is selected, based upon the sample size. In cases where no class can be selected, we set the points in the region to unclassified. Additional processing can include the use of multiple segmentations and merging of classmap values across multiple regions [McKeown and Perlant 92].

Figures 19 and 21 demonstrate the effects of this refinement process. The blocky appearance in the registered classmaps has been improved using the fine grain intensity patches obtained from the high resolution panchromatic imagery. The unclassified pixels in the refined surface material classmap are indicated in black. One significant improvement generated by the refinement process is illustrated by the correct reassignment of soil pixels to concrete pixels on the General Accounting Office Building's top level roof as shown in Figures 17 and 21. Ground features, such as grass areas and asphalt streets, have been diminished in area or extended into other surface materials. This is illustrated by the tile building, immediately below the General Accounting Office Building, with its border areas of grass reassigned to asphalt and portions of the its rooftop modified to asphalt. These reassignments can be attributed to the difference in view angle between the Daedalus ATM multispectral imagery (nadir viewing) and the aerial panchromatic images (off-nadir viewing).

A more quantitative analysis of improvement requires a highly accurate ground-truth surface material classification. This is a current goal of our work in in assessing accuracy for supervised classification (Section 3.5) as well as in assessing the improvements due to refinement.

5.2. Fusion of Classmap with Disparity Images

Various procedures for merging multisensor and multiresolution satellite image data have been explored by the remote sensing community to create composite images that provide more visual information to the image analyst in an interpretable manner [Chavez, et. al. 91]. Previous composite image generation schemes range from merging Landsat TM and SPOT HRV image data sets [Welch and Ehlers 87] to temperature maps with SPOT HRV panchromatic imagery [Schott 89]. Along these same lines, we have implemented a method of merging a refined surface material classmap with a corresponding refined S1S2 disparity image to generate a composite image, displaying surface material and height information simultaneously.

As previously discussed, the refined classmap encodes the surface material type for each image pixel. The refined disparity maps [McKeown and Perlant 92] contains height information as derived by an area-based stereo system, S1, and a feature-based stereo system, S2 [Hsieh, et. al. 92] The disparity images are encoded such that the brighter the pixel intensity level, the closer the pixel surface is to the viewer. Dark pixels are at or below the ground plane. Black pixels indicate that a consistent or refined height estimate was not possible as a result of merging the results of the stereo systems. For example, in Figures 4 and 6, building rooftops appear light gray to white and ground level appears medium to dark gray.

Our classmap-disparity image merging procedure utilizes a hue-saturation-value color model [Foley and Van Dam 83, Smith 78] to transform color images, defined in terms of red, green and blue (RGB) components, into corresponding hue, saturation and value (HSV) components. To fuse the surface material classmap and disparity image together, the refined surface material classmap is transformed from RGB to HSV space. Next, the refined surface material classmap's V component is replaced by a corresponding refined S1S2 disparity image which has been scaled (i.e. linearly stretched to fill the dynamic range of 0-255 intensity levels). Finally, the inverse transformation from HSV to RGB is performed. The resulting color image contains both surface material type and height information for visual inspection.

The refined surface material classmaps, in Figures 19 and 21, were fused with their respective refined S1S2 disparity images shown in Figures 4 and 6. The resulting surface material-height composite images are illustrated in Figures 20 and 22. The color scheme from the refined surface material classmap has been preserved in the surface material-height images with color brightness indicating closeness to the viewer, analogous to the refined disparity image's brightness coding. For example, bright asphalt, tile and concrete features support the presence of a building rooftop while dark grass and asphalt features indicate lawn areas and streets at ground level, respectively. The Interior Department Building, in Figure 20, with its erroneous soil features alerts the need for modification of the surface material classification.

Utilizing the refined S1S2 disparity image for height information, a three-dimensional perspective view of the refined surface material scene can be generated. Perspective views of CIVIL1 and GAO1 refined surface material classmaps are shown in Figures 24 and 26. The color scheme from the refined surface material classmap has been preserved in the three-dimensional scene. Figures 25 and 27 are also three-dimensional refined surface material scene, identical to Figures 24 and 26, but with the image texture incorporated to improve scene structure. Both the surface material-height image and the perspective views of the refined

surface material classmap provide visualizations of scene information. This is a useful starting point for interactive or automated verification and modification of the surface material classmap and the stereo disparity estimates.

Beyond visualization, we are interested in using the surface material class to aid in the refinement and interpretation of the disparity information. We believe that the coupling of surface material classification will allow us to move toward an automated interpretation of the scene in terms of objects such as buildings, trees, and roads. Ambiguities such as an asphalt surface road adjacent to an asphalt roofed building can be disambiguated using height information. Similarly, errors in height information can be detected by looking for implausible material information. For example, isolated tree stands when viewed purely from a height standpoint might be subject to noise processing based upon their lack of regular structure. Given that we can also acquire spectral information consistent with trees or vegetation we may be able to recognize such features directly.

6. Conclusions

Multispectral imagery has provided valuable information to the remote sensing community for the past 20 years. In this paper, we discuss opportunities for the image understanding community to utilize multispectral information in conjunction with high resolution panchromatic imagery. We provide a brief overview of spectral properties for the two major satellite multispectral systems, Landsat TM and SPOT HRV, and describe an airborne multispectral system, the Daedalus ATM. A brief discussion of manual multispectral photointerpretation and automated classification techniques were presented including applications in many disciplines such as forestry, agricultural, regional planning, and wildlife management.

We described a supervised classification procedure, MBCLASS, for multispectral imagery using the Daedalus ATM scanner data over Washington, D.C.. The goal is to automatically generate surface material classmaps that can be used in conjunction with monocular and stereo scene models generated from higher resolution panchromatic aerial imagery.

Finally, two new research areas were presented: refinement of coarse multispectral classmaps and the fusion of surface material classmaps with stereo disparity images. These two areas show that multispectral imagery can provide significant additional scene information to support cartographic applications using image understanding systems and techniques.

In recent years, United States government organizations and professional remote sensing societies have proposed multispectral mapping satellite systems to provide high spatial resolution (10 meter or finer) stereoscopic and multispectral coverage of the Earth's surface [Colvocoresses 90, Light 90]. We believe that the ability to collect multispectral imagery at these spatial resolutions provides the scale needed to enable the application of spatial and structural analysis to complex scenes in urban/suburban environments.

Acknowledgements

We thank SPOT Image Corporation for supplying the Daedalus ATM multispectral imagery flown in the 1983 SPOT Simulation Campaign. Tom Ory of Daedalus Enterprises, Inc. was very helpful in supplying information concerning the Daedalus sensor characteristics and image format. Figure 1 is partially derived from a Daedalus Scanner Application Guide.

We also thank members of the Digital Mapping Laboratory for providing an interesting and weapons-free research environment. Particular thanks to Jeff McMahill for software support in the multispectral analysis and evaluation tools, and to Jefferey Shufelt and Steven Cochran for valuable comments on an earlier draft of this paper.

References

[Ahern, et. al. 91]
Ahern, F. J., Sirois, J., McColl, W. D., Gauthier, R. P., Alfoldi, T. T., Patterson, W. H., and Erdle, T. A.
Progress Toward Improving Aerial Defoliation Survey Methods by Using Electronic Imagers.
Photogrammetric Engineering and Remote Sensing 57(2):187-193, February, 1991.

[Aronoff 82]
Aronoff, S.
Classification Accuracy: A User Approach.
Photogrammetric Engineering and Remote Sensing 48(8):1299-1307, 1982.

[Baumann 90]
Baumann, P. R.
Attempting to Detect and Record Brushland in the Northeastern United States Using MSS Data: Schoharie County, N. Y., as a Case Study.
Photogrammetric Engineering and Remote Sensing 56(3):387-391, March, 1990.

[Chavez, et. al. 91]
Chavez, P. S., Sides, S. C., and Anderson, J. A.
Comparison of Three Different Methods to Merge Multiresolution and Multispectral Data: Landsat TM and SPOT Panchromatic.
Photogrammetric Engineering and Remote Sensing 57(3):295-303, March, 1991.

[Chevrel, et. al. 81]
Chevrel, M., Courtis, M., and Weill, G.
The SPOT Satellite Remote Sensing Mission.
Photogrammetric Engineering and Remote Sensing 47(8):1163-1171, August, 1981.

[Colvocoresses 90]
Colvocoresses, A. P.
An Operational Earth Mapping and Monitoring Satellite System: A Proposal for Landsat 7.
Photogrammetric Engineering and Remote Sensing 56(5):569-571, May, 1990.

[Congalton 88]
Congalton, R. G.
A Comparison of Sampling Schemes Used in Generating Error Matrices for Assessing the Accuracy of Maps Generated from Remotely Sensed Data.
Photogrammetric Engineering and Remote Sensing 54(5):593-600, May, 1988.

[Congalton and Mead 86]
Congalton, R. and Mead, R.
A Review of Three Discrete Multivariate Analysis Techniques Used in Assessing the Accuracy of Remotely Sensed Data from Error Matrices.
IEEE Transactions on Geoscience and Remote Sensing GE-24(1):169-174, January, 1986.

[Congalton, et. al. 83]
Congalton, R., Oderwald, R., and Mead, R.
Assessing Landsat Classification Accuracy Using Discrete Multivariate Analysis Statistical Techniques.
Photogrammetric Engineering and Remote Sensing 49(12):1671-1678, 1983.

[Congalton, et. al. 84]
Congalton, R., Oderwald, R., and Mead, R.
Erratum: Assessing Landsat Classification Accuracy Using Discrete Multivariate Analysis Statistical Techniques.
Photogrammetric Engineering and Remote Sensing 50(10):1477, October, 1984.

[Courtois and Traizet 86]
Courtois, M. and Traizet, M.
The SPOT Satellites: From SPOT 1 to SPOT 4.
Geocarto International 1(3):4-14, 1986.

[Daedalus 83]
Daedalus Enterprise Inc.
Daedalus Airborne Thematic Mapper (ATM AADS 1268)
Daedalus Enterprise Inc, PO Box 1869, Ann Arbor, MI 48106, 1983.

[Dicks and Lo 90]
Dicks, S. E. and Lo, T. H. C.
Evaluation of Thematic Map Accuracy in a Land-Use and Land-Cover Mapping Program.
Photogrammetric Engineering and Remote Sensing 56(9):1247-1252, September, 1990.

[Duda and Hart 73]
Duda, R. O. and Hart, P. E.
Pattern Classification and Scene Analysis.
Wiley, New York, NY, 1973.

[Ehlers, et. al. 90]
Ehlers, M., Jadkowski, M. A., Howard, R. R., and Brostuen, D. E.
Application of SPOT Data for Regional Growth Analysis and Local Planning.
Photogrammetric Engineering and Remote Sensing 56(2):175-180, February, 1990.

[Engel and Weinstein 83]
Engel, J. L. and Weinstein, O.
The Thematic Mapper - An Overview.
IEEE Transactions on Geoscience and Remote Sensing GE-21(3):258-265, July, 1983.

[Fitzpatrick-Lins 81]
Fitzpatrick-Lins, K.
Comparison of Sampling Procedures and Data Analysis for a Land-Use and Land-Cover Map.
Photogrammetric Engineering and Remote Sensing 47(3):343-351, 1981.

[Foley and Van Dam 83]
Foley, J. D. and Van Dam, A.
Intensity and Color.
Addison-Wesley Systems Programming Series. Fundamentals of Interactive Computer Graphics.
Addison-Wesley Publishing Company, Inc., Reading, MA, 1983, pages 613-616, Chapter 17.

[Gong and Howarth 90]
Gong, P. and Howarth, P. J.
An Assessment of Some Factors Influencing Multispectral Land-Cover Classification.
Photogrammetric Engineering and Remote Sensing 56(5):597-603, May, 1990.

[Goodenough, et. al. 87]
Goodenough, D.G., Goldberg, M., Plunkett, G., and Zelek, J.
An Expert System for Remote Sensing.
IEEE Transactions on Geoscience and Remote Sensing GE-25(3):349-359, May, 1987.

[Hodgson, et. al. 87]
Hodgson, M. E., Jensen, J. R., Mackey, H. E., and Coulter, M. C.
Remote Sensing of Wetland Habitat: A Wood Stork Example.
Photogrammetric Engineering and Remote Sensing 53(8):1075-1080, August, 1987.

[Hoffer 78]
Hoffer, R. M.
Biological and Physical Considerations in Applying Computer-Aided Analysis Techniques to Remote Sensor Data.
Remote Sensing: The Quantitative Approach.
In Swain, P. H. and Davis, S. M.,
McGraw-Hill, Inc., New York, NY, 1978, pages 227-289, Chapter 5.

[Hopkins, et. al. 88]
Hopkins, P. F., MacLean, A. L., and Lillesand, T. M.
Assessment of Thematic Mapper Imagery for Forestry Applications under Lake States Conditions.
Photogrammetric Engineering and Remote Sensing 54(1):61-68, January, 1988.

[Hsieh, et. al. 92]
Hsieh, Y.C., McKeown, D.M., and Perlant, F.P.
Performance Evaluation of Scene Registration and Stereo Matching for Cartographic Feature Extraction.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-14(1):to appear, January, 1992.

[Jensen 86]
Jensen, J. R.
Introductory Digital Image Processing: A Remote Sensing Perspective.
Prentice-Hall, Englewood Cliffs, NJ, 1986.

[Kailath 67]
Kailath, T.
The Divergence and Bhattacharyya Distance Measures in Signal Selection.
IEEE Transaction on Communications Technology 15:52-60, 1967.

[Light 90]
Light, D. L.
Characteristics of Remote Sensors for Mapping and Earth Science Applications.
Photogrammetric Engineering and Remote Sensing 56(12):1613-1623, December, 1990.

[Mausel, et. al. 90]
Mausel, P. W., Kramber, W. J., and Lee, J. K.
Optimum Band Selection for Supervised Classification of Multispectral Data.
Photogrammetric Engineering and Remote Sensing 56(1):55-60, January, 1990.

[McKeown 84]
McKeown, D.M.,
Digital Cartography and Photo Interpretation from a Database Viewpoint.
In Gargarin, G. and Golembe, E. (editor), *New Applications of Databases*, pages 19-42.
Academic Press, New York, N. Y., 1984.

[McKeown 87]
McKeown, D.M.
The Role of Artificial Intelligence in the Integration of Remotely Sensed Data with Geographic Information Systems.
IEEE Transactions on Geoscience and Remote Sensing GE-25(3):330-348, May, 1987.

[McKeown and Perlant 92]
McKeown, D., and Perlant, F.
Refinement of Disparity Estimates Through the Fusion of Monocular Image Segmentations.
In *Proceedings of the DARPA Image Understanding Workshop*. Morgan Kaufmann Publishers, Inc., January, 1992.

[NASA 84]
NASA.
A Propectus for Thematic Mapper Research in the Earth Sciences.
Technical Memorandum 86149, Goddard Space Flight Center, Greenbelt, MD, 1984.

[Ormsby and Lunetta 87]
Ormsby, J. P. and Lunetta, R. S.
Whitetail Deer Food Availability Maps from Thematic Mapper Data.
Photogrammetric Engineering and Remote Sensing 53(11):1585-1589, November, 1987.

[Perlant and McKeown 90]
Perlant, F. P. and McKeown, D. M.
Scene Registration in Aerial Image Analysis.
Photogrammetric Engineering and Remote Sensing 56(4):481-493, April, 1990.

[Richards 86]
Richards, J. A.
Remote Sensing Digital Image Analysis: An Introduction.
Springer-Verlag, Berlin, 1986.

[Sabins 87]
Sabins, F. F.
Remote Sensing: Principles and Interpretation.
W. H. Freeman, New York, NY, 1987.

[Schott 89]
Schott, J. R.
Image Processing of Thermal Infrared Images.
Photogrammetric Engineering and Remote Sensing 55(9):1311-1321, September, 1989.

[Slater 80]
Slater, P. N.
Remote Sensing: Optics and Optical Systems.
Addison-Wesley Publishing Company, Reading, MA, 1980.

[Smith 78]
Smith, A. R.
Color Gamut Transform Pairs.
Computer Graphics 12(3):12-19, August, 1978.

[SPOT 83]
SPOT IMAGE Corporation.
1983 U.S. SPOT Simulation Campaign Auxilliary Information Package
SPOT IMAGE Corporation, 1150 17th Street NW Suite 307, Washington, DC 20036, 1983.

[SPOT 84]
SPOT Image Corporation.
SPOT Simulation Applications Handbook.
American Society for Photogrammetry, Falls Church, VA, 1984.

[Story and Congalton 86]
Story, M. and Congalton, R.
Accuracy Assessment: A User's Perspective.
Photogrammetric Engineering and Remote Sensing 53(3):397-399, March, 1986.

[Swain and Davis 78]
Swain, P. H. and Davis, S. M.
Remote Sensing: The Quantitative Approach.
McGraw-Hill, New York, NY, 1978.

[Swain and King 73]
Swain, P. H. and King, R. C.
Two Effective Feature Selection Criteria for Multispectral Remote Sensing.
In *Proc. First International Joint Conference on Pattern Recognition*, pages 536-540. IEEE, Piscataway, NJ, November, 1973.

[Townsend 86]
Townsend, F. E.
The Enhancement of Computer Classification by Logical Smoothing.
Photogrammetric Engineering and Remote Sensing 52(2):213-221, February, 1986.

[Welch and Ehlers 87]
Welch, R. and Ehlers, M.
Merging Multiresolution SPOT HRV and LANDSAT TM Data.
Photogrammetric Engineering and Remote Sensing 53(3):301-303, March, 1987.

[Wharton 82]
Wharton, S. W.
A Context-Based Land Use Classification Algorithm for High Resolution Remotely Sensed Data.
Journal of Applied Photographic Engineering 8(2):46-50, 1982.

[Williams, et. al. 87]
Williams, V. L., Philipson, W. R., and Philpot, W. D.
Identifying Vegetable Crops with Landsat Thematic Mapper Data.
Photogrammetric Engineering and Remote Sensing 53(2):187-191, February, 1987.

[Williamson 89]
Williamson, H. D.
The Discrimination of Irrigated Orchard and Vine Crops Using Remotely Sensed Data.
Photogrammetric Engineering and Remote Sensing 55(1):77-82, January, 1989.

The Direct Computation of Height from Shading

Yvan G. Leclerc and Aaron F. Bobick[1]

Artificial Intelligence Center, SRI International
333 Ravenswood Ave., Menlo Park, CA 94025
(leclerc@ai.sri.com bobick@ai.sri.com)

Abstract

We present a method of recovering shape from shading that solves directly for the surface height. By using a discrete formulation of the problem, we are able to achieve good convergence behavior by employing numerical solution techniques more powerful than gradient descent methods derived from variational calculus. Because we directly solve for height, we avoid the problem of finding an integrable surface maximally consistent with surface orientation. Furthermore, since we do not need additional constraints to make the problem well posed, we use a smoothness constraint only to drive the system towards a good solution; the weight of the smoothness term is eventually reduced to near zero. Also, by solving directly for height, we can use stereo processing to provide initial and boundary conditions. Our shape from shading technique, as well as its relation to stereo, is demonstrated on both synthetic and real imagery.

1 Introduction

The problem of extracting shape from the shaded image of a surface has received considerable attention; an excellent survey is presented in Horn (1990). However, the computation of shape from shading has been typically characterized as that of finding surface orientation, rather than surface height. Converting orientation information into height, or integrating shading methods with other techniques for determining shape, has been less well considered.

In this paper we develop a direct method of computing height from shading. Solving for height, as opposed to orientation, has at least two advantages: First, we do not need to include additional constraints to ensure integrability; any solution necessarily corresponds to a real surface. Second, a formulation expressed in terms of height is more naturally integrated with other methods of recovering shape, such as stereo processing.

[1]The work reported here and the use of the Connection Machine(tm) were partially supported by the Defense Advanced Research Projects Agency. A variant of this paper has been published (Leclerc and Bobick 1991).

We first derive a discrete formulation of the shape from shading problem, and present a solution method that uses a continually decreasing smoothness term to drive the system to a good solution. Specifically, we find the smoothest surface giving rise to the input image. A simple extension allows us to solve for albedo and light source direction. Next, we consider the integration of this height from shading technique with other methods of determining shape. In particular, we employ stereo processing to provide initial conditions for the shading analysis. We also note that stereo and shading are complementary techniques: regions in the image where stereo fails because of the lack of interesting visual events are good candidate regions for shading analysis. We demonstrate our approach on both synthetic and real imagery.

2 Orientation versus Height

2.1 Recovering Orientation

The basic assumption underlying all approaches to shape from shading is the *image irradiance* equation:

$$I(x,y) = R(\vec{n}(x,y)) \qquad (1)$$

which states that image intensity I at a point (x,y) is a function R of the surface normal $\vec{n}$ at the point on a surface that projects to (x,y) in the image. Note that the function R typically contains other variables such as viewer direction, light source direction, and albedo, all of which are typically assumed to be known. We refer to the image irradiance equation as an assumption because it assumes that only the surface orientation, not the surface position, determines the intensity of the reflected light.

The image irradiance equation allows us to characterize the shape from shading problem as finding a surface $z(x,y)$ such that the surface normals satisfy the equation. However, because Equation 1 is expressed in terms of surface orientation, and not surface height, most formulations of the shape from shading problem have focused on recovering surface orientation at each point. If we specify surface orientation using the parameters (p,q)

representing (z_x, z_y), the first derivative of z with respect to x and y, then we can write the image irradiance equation as:

$$I(x,y) = R(p(x,y), q(x,y))$$

and we can express the shape from shading problem as solving for two functions, $p(x,y)$ and $q(x,y)$, such that the irradiance equation holds. Doing so, however, gives rise to two fundamental difficulties. First, the problem is highly underconstrained. For each point (i,j) in an image there is one data point $I(i,j)$ but two unknowns, $p(i,j)$ and $q(i,j)$. The clearest example of this lack of constraint is shown in Horn (1990). Additional constraints, such as the smoothness of the orientations, are required to select a particular solution. Second, arbitrary functions $p(x,y)$ and $q(x,y)$ will not, in general, correspond to orientations of some continuous and differential surface $z(x,y)$ To do so it must be the case that the cross derivatives are equal: $p_y = q_x$. Often, additional processing is required to generate a surface satisfying this constraint (e.g., (Frankot and Chellappa 1988)).

2.2 Recovering Height

A more direct approach to recovering shape from shading is to directly find a surface $z(x,y)$ that minimizes the photometric error. Doing so removes the problem of finding an integrable surface: the recovered function $z(x,y)$ is a real surface whose surface normals accurately predict the image intensities. The direct recovery of surface height forms the basis of the approach here.

We note that the direct recovery of height has been proposed before by Horn and Brooks (1989) but dismissed as computationally divergent. We believe that this result is due to the method of computation: by invoking the calculus of variations they derive a computational scheme equivalent to gradient descent in many variables. This approach is known to have poor numerical convergence properties. The method we present in the next sections demonstrates a discrete method for directly recovering height from shading.

Recently, Horn (1990) developed an approach that considered solving for three functions simultaneously: $z(x,y)$ was added to the functions $p(x,y)$ and $q(x,y)$. The objective function includes a term $((z_x - p)^2 + (z_y - q)^2)$ which drives the three functions z, p, and q to approximately represent the same real surface. However, the recovered surface z never exactly corresponds to the orientations (p,q) used to compute the photometric error. In the following section, we present a method in which p and q are derived from z, thereby eliminating this source of error.

3 Height from Shading

3.1 Discrete Formulation

Recently, most formulations of the shape from shading problem have been expressed as a problem in the calculus of variations (Horn and Brooks 1989). In this view, the task of shape from shading is one of recovering a function that minimizes a functional. Thus for the case of height from shading, one would seek to recover the function $z(x,y)$. The attraction of this approach is that it provides an elegant framework in which to describe the shading problem and in which to derive necessary conditions that constrain the solutions. In particular, Euler's equation expresses a necessary condition in terms of the derivatives of the function $z(x,y)$. This condition can then be manipulated into an iterative solution method.

Unfortunately, the iteration equations derived from Euler's equation yield methods equivalent to gradient descent. For example, the difference equations in Brooks and Horn (1985) can be derived by taking the gradient of their objective function with respect to each z_{ij}. Gradient descent algorithms are known to have poor convergence properties for systems of many variables. To avoid this problem we formulate a shape from shading objective function in a purely discrete manner, permitting us to consider the problem as one of ordinary calculus in many variables (see (Szeliski 1991)). This approach allows the use of numerical solution methods with better convergence properties.

Here we present our discrete formulation of an objective function to be minimized in solving the shape from shading problem. As mentioned in the previous section, our method of computing height from shading initially drives the computation using a smoothness term. Thus our objective function contains a smoothness term along with a photometric error term:

$$E = \sum_{i,j} (1-\lambda)(R(p_{ij}, q_{ij}) - I_{ij})^2 + \lambda(u_{ij}^2 + v_{ij}^2) \quad (2)$$

where p, q, u, v are not independent variables but are defined as the symmetric first and second finite differences of the variables $\vec{z} = (z_{ij})$:

$$\begin{aligned} p_{ij} &= \frac{1}{2}(z_{i+1,j} - z_{i-1,j}) \\ q_{ij} &= \frac{1}{2}(z_{i,j+1} - z_{i,j-1}) \\ u_{ij} &= z_{i+1,j} - 2z_{i,j} + z_{i-1,j} \\ v_{ij} &= z_{i,j+1} - 2z_{i,j} + z_{i,j-1} \end{aligned}$$

λ represents a continuation parameter, $0 \leq \lambda \leq 1$, that is gradually decreased to near zero. The particular smoothness term above represents deviation from a plane, and may in fact be too restrictive for some cases, even for small λ. We are currently considering a smoothness term

that measures variation in curvature as opposed to orientation.

Note that so far we have not discussed the reflectance function R; to make our derivation explicit it is essential to choose some particular R. The results presented here employ a Lambertian shading model where:

$$R_{ij} = R(p_{ij}, q_{ij}) = \vec{\mathbf{n}}_{ij} \cdot \vec{\mathbf{l}} = \frac{ap_{ij} + bq_{ij} - c}{\sqrt{1 + p_{ij}^2 + q_{ij}^2}}$$

where $\vec{\mathbf{n}}_{ij}$ is the unit vector surface normal, and $\vec{\mathbf{l}} = \langle a, b, c \rangle$ is the unit light source vector scaled by the albedo. For now we assume the scaled light source vector is known; in the next section we consider solving for $\vec{\mathbf{l}}$.

Given the objective function expressed in Equation 2, we can derive the gradient of E: a vector whose elements are the partial derivatives of E with respect to the state variables z_{ij}. First, define N_{ij} and $\sqrt{D_{i,j}}$ to be the numerator and denominator of R_{ij} above. Then, the elements of the gradient are

$$\begin{aligned}
\frac{\partial E}{\partial z_{ij}} &= (1-\lambda) \times \qquad (3)\\
&\left\{ \frac{R_{i-1,j} - I_{i-1,j}}{\sqrt{D_{i-1,j}}} \left(a - \frac{N_{i-1,j}}{2D_{i-1,j}} p_{i-1,j} \right) \right.\\
&+ \frac{R_{i+1,j} - I_{i+1,j}}{\sqrt{D_{i+1,j}}} \left(-a + \frac{N_{i+1,j}}{2D_{i+1,j}} p_{i+1,j} \right)\\
&+ \frac{R_{i,j-1} - I_{i,j-1}}{\sqrt{D_{i,j-1}}} \left(b - \frac{N_{i,j-1}}{2D_{i,j-1}} q_{i,j-1} \right)\\
&\left. + \frac{R_{i,j+1} - I_{i,j+1}}{\sqrt{D_{i,j+1}}} \left(-b + \frac{N_{i,j+1}}{2D_{i,j+1}} q_{i,j+1} \right) \right\}\\
&+ 2\lambda \{ -2u_{ij} - 2v_{ij} + u_{i+1,j}\\
&\quad + u_{i-1,j} + v_{i,j+1} + v_{i,j-1}) \}
\end{aligned}$$

Explicit derivation of the gradient is essential if we are to make use of more powerful numerical methods for minimizing the objective function, as discussed in Section 4.

3.2 Source Direction and Albedo

Though the above discussion assumes a known light source direction and albedo $\vec{\mathbf{l}} = \langle a, b, c \rangle$ we can also consider minimizing E with respect to these parameters, as did Horn and Brooks (1985) within their variational formulation. In fact, the processing of real imagery usually requires such estimation since light source direction and albedo are rarely known accurately. Furthermore, one can show that the objective function is highly sensitive to errors in albedo. To avoid this problem we explicitly solve for the albedo as well as the light source direction.

Following our approach we need to construct the gradient of E with respect to $\vec{\mathbf{l}}$:

$$\begin{aligned}
\frac{\partial E}{\partial a} &= 2 \sum_{ij} (1-\lambda)(R_{ij} - I_{ij}) \frac{p_{ij}}{\sqrt{D_{i,j}}}\\
\frac{\partial E}{\partial b} &= 2 \sum_{ij} (1-\lambda)(R_{ij} - I_{ij}) \frac{q_{ij}}{\sqrt{D_{i,j}}} \qquad (4)\\
\frac{\partial E}{\partial c} &= 2 \sum_{ij} (1-\lambda)(R_{ij} - I_{ij}) \frac{-1}{\sqrt{D_{i,j}}}
\end{aligned}$$

Using these equations we can simply consider the scaled light source vector as just another set of variables; doing so has in practice yielded good results (see sections 5 and 6). Note that the above is a set of three simultaneous linear equations in a, b, and c, and could be solved directly when $\vec{\mathbf{z}}$ is known.

3.3 Existence and Uniqueness

Whether we formulate shape from shading as a problem of differential equations or as one of minimization, we should consider the issue of the existence of a solution and whether a solution is unique. Results in the continuous domain (Blake *et al.* 1985, Bruss 1982) have shown a unique solution to the shape from shading problem under the restrictive conditions that the light source direction is equal to the viewing direction, boundary conditions are completely specified, and that the data are noise free. Recently, Oliensis (1991) showed that if there is at least one visible point on the surface whose normal is parallel to the light source, then there is a unique surface corresponding to the image. With respect to existence, Horn, Szeliski, and Yuille (1990) recently demonstrated some impossible shaded images that cannot be generated by shading a surface with continuous first derivatives.

However, any computational solution to the shape from shading problem has three characteristics that make existence and uniqueness difficult to consider. First, using a discrete formulation with quantized variables makes some continuous analysis inapplicable. For example, boundary conditions no longer can propagate arbitrarily far. Second, real images must be considered as discrete samplings of underlying continuous images, and the method of computing discrete derivatives can greatly affect predicted image intensities for a given $\vec{\mathbf{z}}$. Third, and perhaps most important, real images have noise, whether it is real noise induced by the imaging system, quantization of image intensities, or deviations from the assumed reflectance function. In such circumstances, it is usually possible to construct multiple solutions whose objective function measures are approximately equal.

An example of the difficulties inherent in discrete formulations is shown in Figure 1 which displays Lambertian shaded images of two synthetic surfaces, illuminated from two different light source directions. The top row shows a pair of intersecting hemispheres illuminated

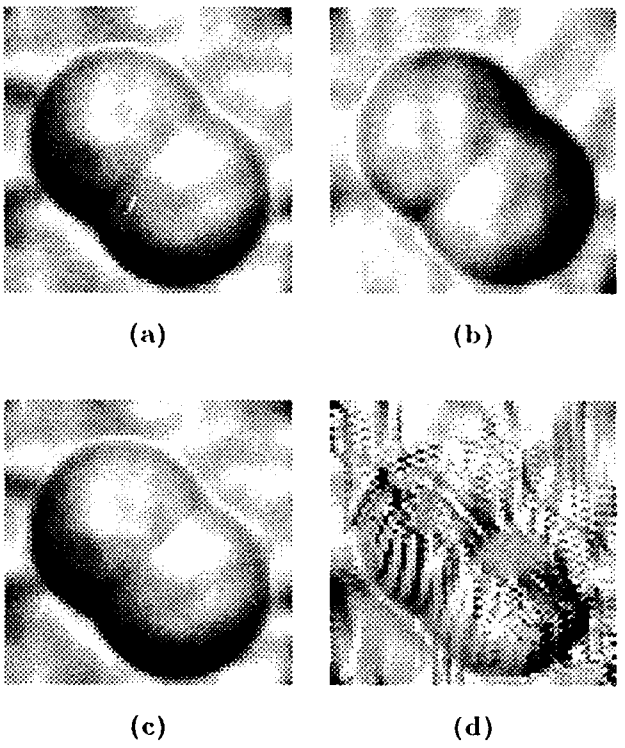


Figure 1: Demonstration of the nonuniqueness of shape from shading when there are no boundary conditions, allowing for a small, but nonzero, photometric error. These, and all other synthetic images in this paper, are produced by shading a height map, $\vec{z} = \langle z_{ij} \rangle$. (a,b) Two images of a surface consisting of two bumpy hemispheres, shaded from the northeast and northwest. (c,d) Two images of a very different, wrinkled surface, also shaded from the northeast and northwest. When shaded from the northeast, this surface looks the same as the hemispheres: (a) and (c) have an average absolute difference in intensity of less than 0.1%. When shaded from the northwest, however, we see the tremendous differences between the two. In short, very different surfaces can look the same under identical viewing conditions.

from the northeast and the northwest. The bottom row shows a rough oscillating surface. When shaded in exactly the same northeast direction as the hemispheres, the wrinkled surface yields an image quite similar to the image above it, with an average difference of less than 0.1%. Thus, if the top left image is taken as a data image, then the surface of the bottom row will produce a low photometric error, and hence a low value for E in Equation 2 whenever the smoothness weight λ is near zero. The space of these undesired solutions is determined by the definitions of the discrete derivatives of $\vec{z}$. Note that these solution are real surfaces and that the ambiguity cannot be attributed to the nonintegrability of surface orientations.

The implication of this demonstration is that when computing a surface that satisfies a shaded image, additional constraints must be imposed by the solution method. In the sections that follow we drive the system with an initial smoothness term in order to select as smooth a surface as possible with low photometric error. Whether this is the preferred solution is unclear: are smooth surfaces preferable? We are currently investigating using a general position view of preference to select the solution whose shading changes the least with movement of the light source direction, or with a change in the definition of discrete derivates.

One should be aware that the question of uniqueness becomes even harder to consider if the light source and albedo are allowed to vary. For example, there is a concavity/convexity ambiguity depending on whether the light source is seen as coming from above or below. Within an optimization framework, a poor choice of solution method and initial condition can yield multiple solutions. For complicated synthetic or realistic images where there are enough variations in orientation we have not seen multiple solutions for the light source direction.

4 Solution Method

Our goal is to find a surface that is consistent with the shaded image. When there is no noise, this means finding a surface that incurs zero photometric error. However, as seen in Figure 1, there are typically many surfaces that have $E \approx 0$. Which of these surfaces should we choose? Our criterion is to choose the smoothest such surface.

To find the smoothest surface that incurs zero photometric error, we use a continuation method (Leclerc 1989), whereby we begin by finding a local minimum of Equation 2 for $\lambda = 1$. Once a minimum is found, λ is decreased, and a search for a minimum of this new objective function is begun, using the previous minimum as initial condition. This procedure is repeated until λ is sufficiently close to zero. When there is no noise, the theoretical limit for λ is exactly zero. In the presence of noise, the appropriate final value for λ depends on the extent of the noise; for nonzero λ, the global minimum value of E is, in general, also nonzero.

Our solution method implements the standard conjugate gradient algorithm FRPRMN (from (Pre... *et al.* 1986)) in conjunction with the line search algorithm DBRENT as an iterative minimization technique. This algorithm simply requires the construction of two functions: one that computes the value of E, Equation 2, and one that computes the gradient of E with respect to the state vector $\vec{z}$, Equation 4. To impose boundary conditions, i.e., fix some elements of $\vec{z}$ to their known correct values, we define the gradient to be zero at those points. We use a conjugate gradient technique rather than simpler gradient descent algorithms because the former is a much more efficient technique for optimizing functions

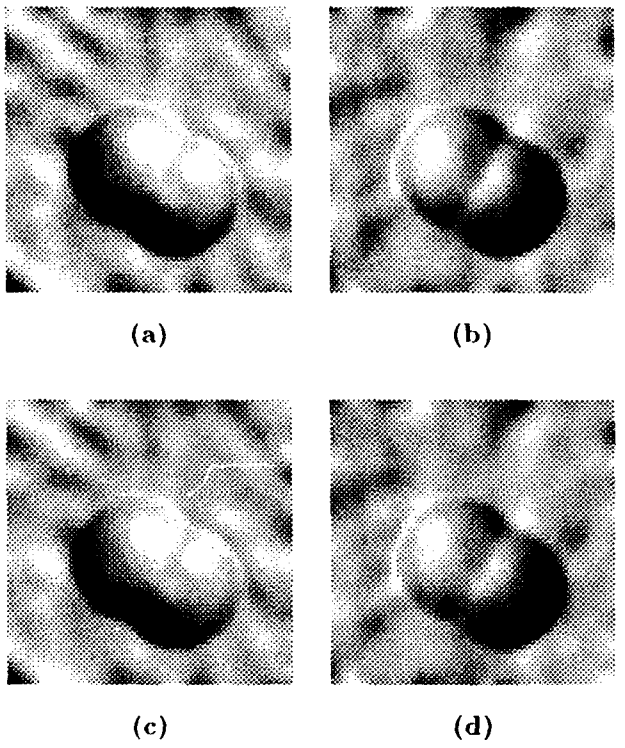


Figure 2: Result using hierarchical continuation method on a similar input image to that of Figure 1, without boundary conditions. (a) Input image of true surface. (b) Image of true surface illuminated with light source at 90° from (a). (c) Image of recovered surface using original light source direction. (d) Image of recovered surface using rotated light source.

of many variables (Szeliski 1991).

Given that the solution found at the first step is very smooth, we would expect that solutions found at each subsequent step would also be relatively smooth, within the constraints imposed by the image data. Although we cannot guarantee that the solution found for a small λ is indeed the smoothest surface possible, experiments demonstrate that the solutions recovered are indeed smooth surfaces incurring small photometric error.

Figures 1 and 2 illustrate the above approach. For Figure 1, the system began with $\lambda = 1/16$, while for Figure 2, the system started with $\lambda = 1$. In both cases, the average photometric error is very small: the average absolute difference between the images of Figures 1(a) and 1(c) and between those of 2(a) and 2(c) is less than 0.1%. However, the recovered surface for Figure 2 is much smoother, as we can see by the image in the lower right. Indeed, the recovered surface corresponds to that of the true surface—the surface used to generate the original synthetic image—to within less than 2% average absolute difference. In this particular example, no boundary conditions were imposed, yet the solution recovered is correct. The conjugate gradient algorithm guarantees that the system is stable whether or not boundary conditions are imposed. We are investigating the conditions under which boundary conditions are required to recover the correct solution.

Another difference in the processing of Figures 1 and 2, is that the latter was produced using a hierarchical technique (Terzopoulos 1983). Specifically, the original image is initially blurred and subsampled from its initial resolution, in this case, 64×64, down to a resolution of 8×8. The system begins with $\lambda = 1$, which is progressively reduced to $1/16$ by a factor of $1/\sqrt{2}$. The resultant surface is then bilinearly interpolated to the next higher resolution, 16×16, and the process begun anew, but with starting and ending λ equal to half of those of the previous resolution. At each stage, the input image is an appropriately blurred and subsampled version of the original image. This procedure is repeated to the full resolution of the initial image. Even though blurring and subsampling the original image does not generate exactly the same image as shading the blurred and subsampled surface (Ron and Peleg 1989), it appears to be a sufficiently good starting point for the minimization. Using the hierarchy can save at least a factor of two in computation time.

Finally, to recover albedo and light source direction, we have employed two strategies. The most direct is to incorporate the light source direction parameters into the state variable vector and to solve for $\vec{z}$ and $\vec{l}$ simultaneously within the conjugate gradient framework. In this case, it is necessary to scale the contribution of the gradient elements of Equation 4 by the number of pixels in the image to make their magnitude commensurate with the other elements of the gradient vector. This procedure has yielded good results for many synthetic images; however, we have found cases where the solution is trapped in poor local minima. The second technique is to use an a priori surface estimate, generated by some other method such as stereo, and to minimize the objective function by varying only the light source parameters. After finding the best solution for the light source, those parameters are fixed, and $\vec{z}$ is then allowed to vary.

5 Stereo and Shading

To this point we have developed a method for extracting shape from shading that directly operates on heights $\vec{z}$ of a surface. As mentioned, one of the advantages of this approach is that it allows for the incorporation of other methods of determining surface height into the shape recovery process. In this section we develop the relationship between stereo processing and shape from shading; it is our belief that the two methods are inherently complementary and will function much more effectively in

an integrated fashion.

The most obvious connection between shading and stereo is that stereo is an explicit method for providing initial and boundary conditions for the shading problem. We believe stereo is particularly appropriate for this task because stereo is sensitive to variations in height, not orientation. This condition results in the linear decrease of the discriminability of relative heights as absolute distance increases. Shading, however, reflects change in orientation, and thus does not lose discrimination power with increasing absolute distance. Given stereo information to coarsely describe surface shape, shading analysis can solve for finer surface variations.

We demonstrate the ability of stereo to provide initial conditions for shading processing in Figures 3 and 4. The top left images (3a and 4a) are each one half of a stereo pair, where regions of significant albedo variation (eyes and nostril in Figure 3 and the background in both) have been manually removed. Displayed at the top right (3b and 4b) are the stereo reconstructed surfaces, computed using Fua's stereo algorithm (Fua 1991) (people acquainted with the subject in Figure 3 could not recognize him from this image). Using the stereo reconstructed surfaces as initial conditions, our shading algorithm recovered the surfaces displayed in the bottom two images of each figure. The images on the bottom left (3c and 4c) are the recovered surfaces shaded from the same direction as the original images. For Figure 3, we first used the stereo solution to solve for the light source direction and albedo, and then allowed the surface to vary. For Figure 4, we measured the light source direction in the laboratory. The bottom-right images (3d and 4d) display the recovered surfaces shaded from a different direction. Though some creases invisible to the light source direction have been introduced, the surfaces capture most of the important shape characteristics including wrinkles in the forehead of Figure 3. The introduction of creases is most likely caused by using a smoothness term that prefers planar patches; creases give large patches of no smoothness penalty plus thin ridges of high penalty. Even though the weight of the smoothness term is eventually reduced to near zero, the system can no longer remove the creases. A higher order smoothness term may solve this problem.

Aside from providing initial and boundary conditions, there is a much deeper relationship between stereo and shading, and that relationship is derived from the conditions under which each of the two methods operate best. Recovery of distance information from stereo processing requires the ability to make accurate matches between corresponding pixels in a stereo pair. Such accurate matches can occur only where there are significant events in the image intensities that can disambiguate pixel matches. Such events include discontinuities in surface orientation and albedo, such as material boundaries and resolvable textures. Where no such events occur, all stereo algorithms determine surface structure by interpolation, whether explicitly (e.g. (Grimson 1982)) or implicitly by choice of objective function (e.g. (Barnard 1989)). Grimson justified this interpolation in stating that "No news is good news," implying that the lack of significant events could be viewed as permitting the use of some assumed interpolation function.

Shading analysis, however, operates best in exactly those regions of an image where stereo processing is forced to interpolate. Thus, "No news is better news" in the sense that lack of significant visual events may be used as an indication that shading analysis is appropriate. One way of viewing this relationship is that shading analysis should provide the interpolation function, as opposed to making some specific assumptions about the most appropriate type of spline, e.g. fractured thin-plates or membranes. One implication of this type of approach is that smooth albedo variations would be incorrectly interpreted as shape information; the effectiveness of makeup to give the illusion of greater depth variation is an example of such an incorrect interpretation.

Many stereo algorithms (Fua 1991, Hannah 1982) already provide a confidence measure reflecting the degree of constraint present in the pixel matches. We are currently investigating the use of this measure as a method for generating an integrated stereo and shading algorithm that applies each where best suited. Essential to this approach is a shape from shading technique based on surface heights, not orientations. The technique developed here should provide the necessary basis.

6 Summary

We have presented a method of recovering shape from shading which solves directly for the surface height. By using a discrete formulation, we are able to employ numerical solution methods more powerful than gradient descent, giving good convergence behavior. Because height is directly recovered, we avoid the problem of finding an integrable surface maximally consistent with surface orientation. Furthermore, since we do not need additional constraints to make the problem well posed, we use a smoothness constraint only to drive the system to a good solution: the smoothest surface that incurs no photometric error. Eventually we remove the smoothness term, preventing the system from walking away from the true solution. Our solution technique uses a continuation method in the smoothness parameter, embedded in a hierarchical conjugate gradient minimization scheme. A simple extension allows for the solution of light source direction and albedo as well as surface height. We have demonstrated this technique on both synthetic and real imagery.

In addition, we have begun to explore the relationship between shading and stereo. In particular, because

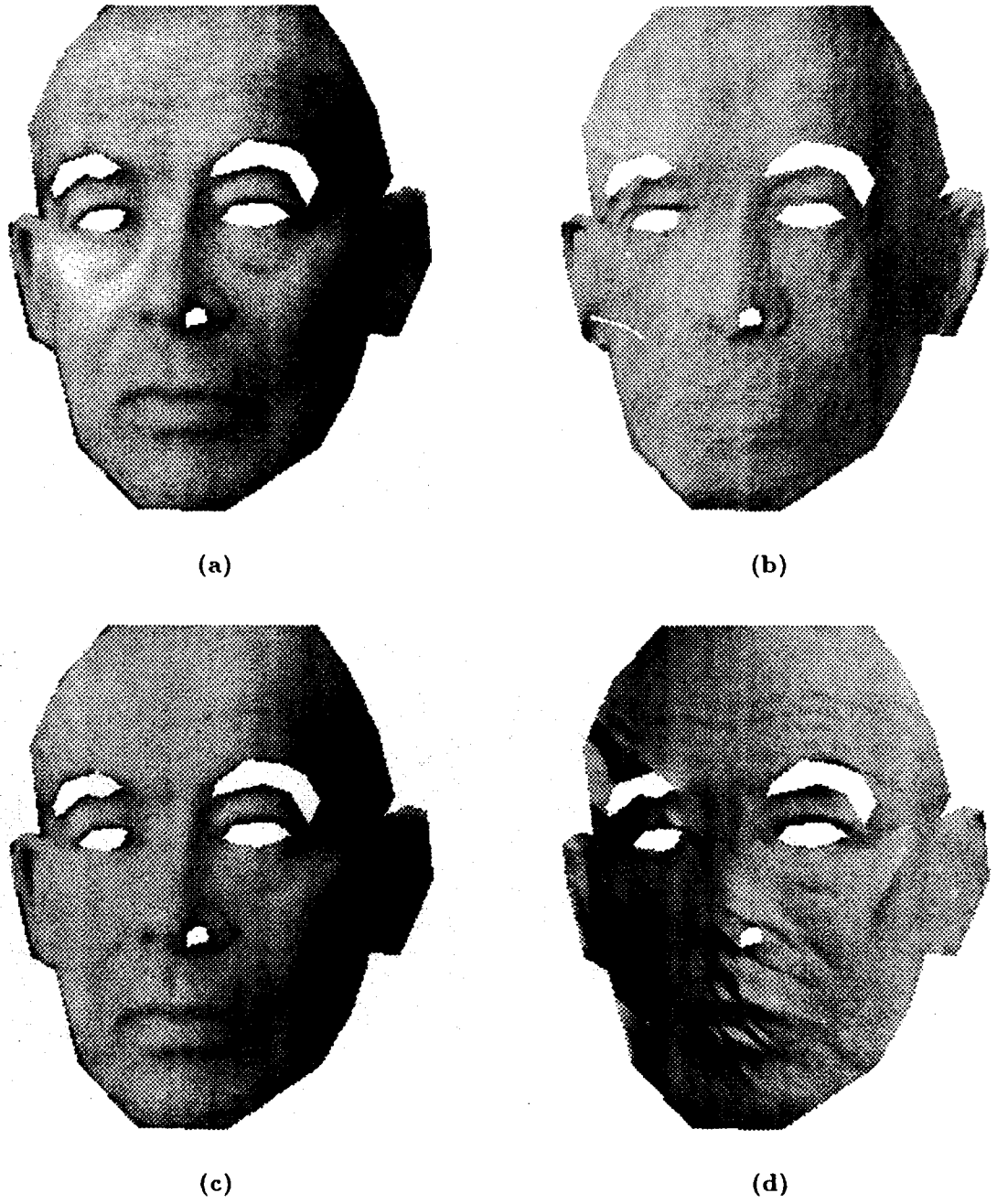


Figure 3: *Four faces of Oscar.* (a) *Grey level image of a face with regions of significant albedo change manually removed. This image is one of a stereo pair.* (b) Surface recovered by stereo processing. Note the coarse resolution. (c) Shaded image of recovered surface when shaded using same light source direction as solved for in (a). (d) Same surface shaded from different direction. Though it contains some creases invisible to original light source direction, the surface captures most of the important shape characteristics, including the wrinkles in the forehead.

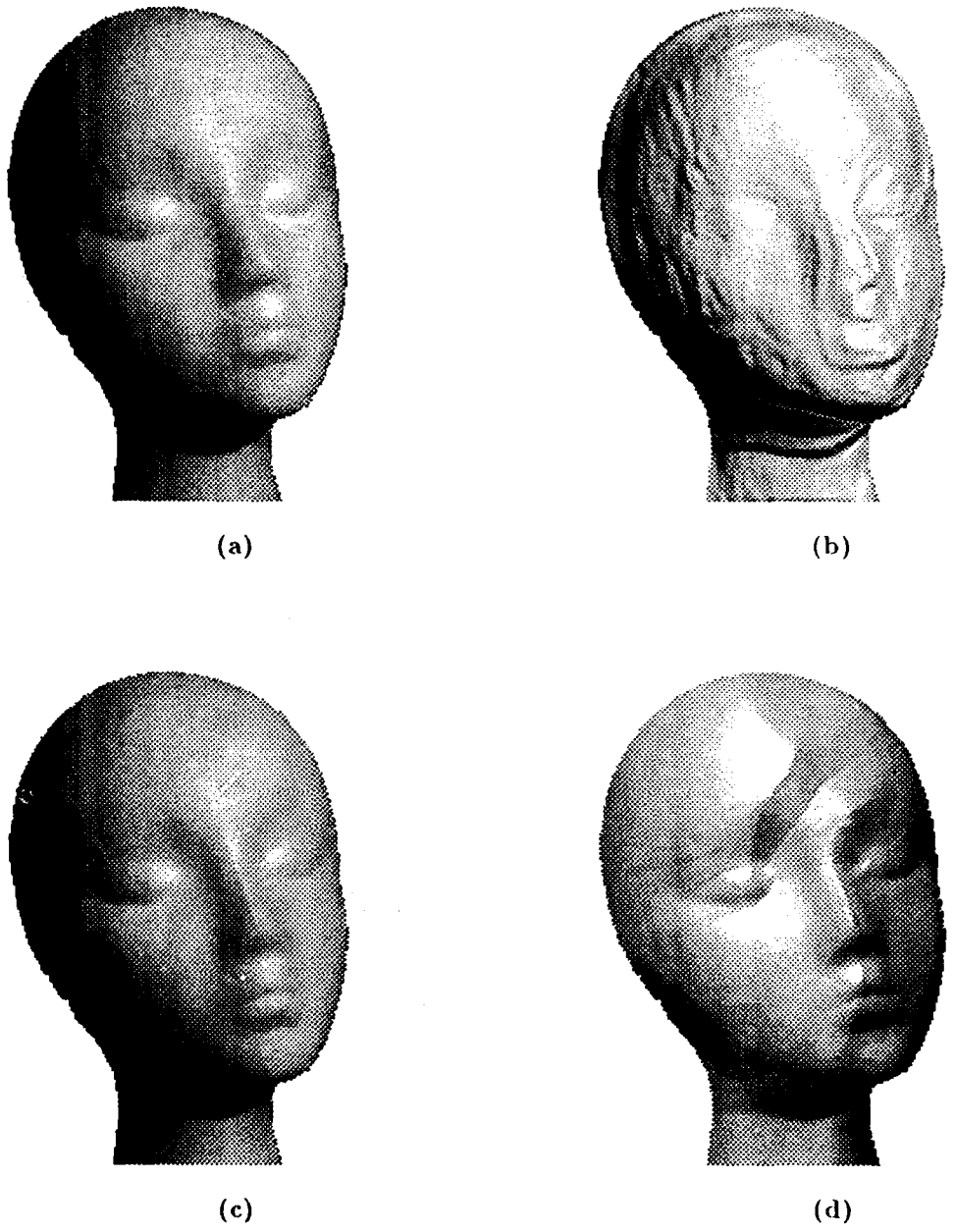


Figure 4: Laboratory experiment on a styrofoam mannequin spray-painted using matte white paint. Panels (a)–(d) are as in Figure 3. In this case, the illuminant direction was measured explicitly, and did not need to be estimated from the image.

we have a formulation in terms of surface height we can use stereo processing to provide initial and boundary conditions. We note that shading and stereo are complementary techniques for two reasons: First, relative depth discrimination from stereo decreases with absolute depth, whereas orientation discrimination, as determined by shading, does not. Second, regions in the image where stereo fails because of lack of interesting visual events are good candidate regions for shading analysis. One goal for future work is to use the confidence measures of stereo systems to invoke the application of our shading analysis and to control the balance between the shading and stereo solutions.

Another extension of this work is to use a piecewise constant albedo model that allows different regions in the image to have different and unknown albedos. An algo-

rithm has been implemented in which only the position of the albedo discontinuities are known; these discontinuities should be recoverable using edge detection when they are the only ones present. This algorithm has performed well on all of our original synthetic examples of constant albedo, and on additional synthetic images of mottled surfaces with two different albedos, much like zebra skin.

References

Barnard, S. (1989). Stochastic stereo matching over scale. *Int'l J. Computer Vision, 3(1)*, 17–32.

Blake, A., Zisserman, A., and Knowles, G. (1985). Surface descriptions from stereo and shading. *Image Vision Comput., 3(4)*, 183–191.

Brooks, M. J. and Horn, B. K. P. (1985). Shape and source from shading. In *Proc. Intern. Joint Conf. Art. Int.*, pages 932–936, Los Angeles.

Bruss, A. R. (1982). The eikonal equaion: some results applicable to computer vision. *J. Mathematical Physics, 23(5)*, 890–896.

Frankot, R. T. and Chellappa, R. (1988). A method for enforcing integrability in shade from shading algorithms. *IEEE Trans. PAMI, 10(4)*, 439–451.

Fua, P. (1991). *A parallel stereo algorithm that produces dense depth maps and preserves image features.* Rapports de Recherche 1369, INRIA-Sophia Antipolis. Submitted to J. Machine Vision and Applications.

Grimson, W. E. L. (1982). A computational theory of visual surface interpolation. *Philosophical Trans. of the Royal Society of London B, 298*, 395–427.

Hannah, M. J. (1982). Bootstrap stereo. In *Proc. DARPA Image Understanding Workshop*, pages 201–208, Palo Alto, CA.

Horn, B. K. P. (1990). Height and gradient from shading. *Int'l J. Computer Vision, 5(1)*, 37–75.

Horn, B. K. P. and Brooks, M. J. (1989). The variational approach to shape from shading. *Computer Vision, Graphics, and Image Processing, 33(2)*, 174–208.

Horn, B. K. P., Szeliski, R., and Yuille, A. L. (1990). Impossible shaded images. *submitted to IEEE Trans. PAMI*, .

Leclerc, Y. G. (1989). Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision, 3(1)*, 73–102.

Leclerc, Y. G. and Bobick, A. F. (1991). The direct computation of height from shading. In *Proceedings of the 1991 Computer Society Conference on Computer Vision and Pattern Recognition*, Lahaina, Maui, Hawaii.

Oliensis, J. (1991). Shape from shading as a partially well-constrained problem. *CVGIP: Image Understanding*, , to appear.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1986). *Numerical methods, the art of scientific computing.* Cambridge U. Press, Cambridge, MA.

Ron, G. and Peleg, S. (1989). Multiresolution shape from shading. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 350–355, San Diego, CA.

Szeliski, R. (1991). Fast shape from shading. *CVGIP: Image Understanding, 53(2)*, 129–153.

Terzopoulos, D. (1983). Multilevel computational processes for visual surface reconstruction. *Computer Vision, Graphics, and Image Processing, 24*, 52–96.

THE INTERPRETATION OF SYNTHETIC APERTURE RADAR IMAGES USING PROJECTIVE INVARIANTS AND DEFORMABLE TEMPLATES

Aaron J. Heller, David M. LaRocque and Joseph L. Mundy *
GE Corporate Research and Development Center
Schenectady, NY 12345

Abstract

In this paper, we discuss a technique for model-based object recognition in Synthetic Aperture Radar (SAR) images which combines projective invariants and deformable templates. The models are comprised of deformable templates whose configuration is constrained by projective invariants. An operator outlines an area of interest which provides initial values for the parameters of the deformable template. The constraints on the template are represented as a system of polynomials derived from the projective invariants. A non-linear system solver then optimizes the alignment of model features and image features subject to the constraints. The technique has been successful in locating geographic features in aerial SAR images with models constructed from USGS maps.

1 Introduction

Images produced by Synthetic Aperture Radar are difficult to interpret – the resolution is poor, the usual cues (shading, shadows, etc.) are missing, and the appearance of an object can change radically with small changes in pose. Image analysts have told us that in some cases it can be difficult just to orient the image correctly. The experiment described here is a "proof of concept" study for the design of an automatic model-based object recognition module for an image analyst's workstation.

2 The Approach

The underlying design of the system combines two model-based techniques for object recognition – deformable templates[Lipson *et al.*, 1990] and projective invariants[Forsyth *et al.*,] [Coelho *et al.*, 1991]. By using deformable templates we avoid having to make a priori assumptions about the correspondence of model features and image features. By constraining the allowable configurations of the deformable templates with projective invariants, we are assured that the configuration of the template is a valid projection of the model.

2.1 Projective Invariants

Shape characteristics measured in images depend not only on the intrinsic properties of an object's shape, but also on the position and orientation of the camera with respect to the object and the intrinsic parameters of the camera (e.g. focal length, frame aspect ratio, etc.). Shape *descriptors* which are unaffected by these projective transformations can be constructed. Such descriptors, known as projective invariants, can be matched to object properties regardless of camera viewpoint and parameters. Essentially, the use of projective invariants factors the object's pose and the camera's imaging parameters out of the recognition process.

In the initial implementation we have employed the five-coplanar points invariant. Other planar invariants suitable for this technique are five-coplanar lines (the dual of five-coplanar points), a pencil of four lines and the joint invariants of two conics /refpami. It is also reasonable to exploit the invariants available from multiple views,e.g., the invariant of nine points observed from two arbitrary perspective views.[1].

Five coplanar points give rise to two invariants which have found important applications in machine vision [Barrett *et al.*, 1991, Mohr and Morin, 1990] and photogrammetry [Moffitt and Mikhail, 1980]. These invariants have been related to the familiar cross ratio that is obtained when the image points fall on a line [Brill and Barrett, 1983].

To derive these projective invariants, consider five points on a plane. We represent these points as $\mathbf{p}_i = (x_i, y_i, z_i)$. Clearly, a change in scale factor $\mathbf{p}_i \rightarrow \lambda \mathbf{p}_i$ cannot be observed and this must be taken into account in deriving the invariants. As in the case of the cross ratio, these invariants are ratios of scalars. Consider a square matrix defined by three of the five points.

$$\mathbf{m}_{ijk} = \begin{pmatrix} x_i & x_j & x_k \\ y_i & y_j & y_k \\ z_i & z_j & z_k \end{pmatrix}$$

Under a general projective transformation T

$$\mathbf{M}_{ijk} = \mathbf{T}\,\mathbf{m}_{ijk}$$

*Work at GE was supported in part by the DARPA Strategic Computing Vision Program and the Air Force Office of Scientific Research under Contract No. F49620-89-C-0033 and by the DARPA Strategic Computing Vision Program under Contract No. MDA972-91-C-0053.

[1] See a review of the nine point invariant and its applications in "3D Model Alignment Without Computing Pose", by J.L. Mundy *et al.* in this proceedings.

It can be shown that there are only two functionally independent ratios of the determinants of the matrices, $\mathbf{M}_{ijk}$, of the five points which are invariant for projective transformations and the homogeneous scalar factor.

$$I_1 = \frac{det(\mathbf{M}_{431})det(\mathbf{M}_{521})}{det(\mathbf{M}_{421})det(\mathbf{M}_{531})}$$

$$= \frac{|\mathbf{T}|(\lambda_4\lambda_3\lambda_1)det(\mathbf{m}_{431})|\mathbf{T}|(\lambda_5\lambda_2\lambda_1)det(\mathbf{m}_{521})}{|\mathbf{T}|(\lambda_4\lambda_2\lambda_1)det(\mathbf{m}_{421})|\mathbf{T}|(\lambda_5\lambda_3\lambda_1)det(\mathbf{m}_{531})}$$

$$= \frac{det(\mathbf{m}_{431})det(\mathbf{m}_{521})}{det(\mathbf{m}_{421})det(\mathbf{m}_{531})}$$

in the same way

$$I_2 = \frac{det(\mathbf{m}_{421})det(\mathbf{m}_{531})}{det(\mathbf{m}_{432})det(\mathbf{m}_{521})}$$

Note that if two of the points triples are collinear the point matrix $\mathbf{m}_{ijk}$ becomes singular and the corresponding invariant is undefined. Since points and lines are dual in the projective plane, we have immediately that these functions are also invariants for a system of five coplanar lines, no three of which are coincident.

2.2 Deformable Templates

Deformable templates can be used to detect features in images[Lipson *et al.*, 1990]. The templates are specified by a set of parameters, in our case model points and the projective invariant relationships among them. This enables *a priori* knowledge about the expected relationship of the image features to guide the detection process. The deformable templates interact with the image data in a dynamic manner. An objective function is defined which "attracts" the template to the image features, which, in this experiment are vertices defined by points of high curvature along edges in image intensity. The maximum of the objective function corresponds to the best fit with the image data. The objective function is given by:

$$f(\mathbf{x}) = \sum_{i \epsilon M} \sum_{j \epsilon D} e^{-r_{ij}^2}$$

Where M is the set of model features, D is the set of image features and r_{ij} is the Euclidean distance from the location of a given image feature, j, to a given template feature, i. Note that:

- No assumption is made about the correspondence between image and model features.
- Since the constraints on the template configuration are projective invariants derived from the model, the template can only take on configurations that correspond to valid projections of the model.

2.3 Solving Constraints

The role of the constraint solver is to solve the problem of finding a configuration of template features which satisfies all of the constraints defined by the model, and agrees as well as possible with the image features.

The two goals of finding the global maximum of the objective function, $\nabla f(\mathbf{x}) = \mathbf{0}$, and satisfying the constraints, $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, are combined to give a constrained minimization problem. A linear approximation to this optimization problem is:

$$\begin{aligned} \nabla^2 f(\mathbf{x})\,\mathbf{dx} &= -\nabla f(\mathbf{x}) \\ \nabla \mathbf{h}(\mathbf{x})\,\mathbf{dx} &= -\mathbf{h}(\mathbf{x}) \end{aligned} \qquad (1)$$

Since the two goals cannot in general be simultaneously satisfied, a least-square-error satisfaction of $\nabla f(\mathbf{x}) = \mathbf{0}$ is sought. The constraint equations are multiplied by a factor $\sqrt{c}$, which determines the weight given to satisfying the constraints versus minimizing the cost function. Each iteration of (1) has a line search that minimizes the least-square-error:

$$m(\mathbf{x}) = |\nabla f(\mathbf{x})|^2 + c\,|\mathbf{h}(\mathbf{x})|^2 \qquad (2)$$

which is a merit function similar to the objective of the standard penalty method.

Starting with $\sqrt{c} = 0$, system (1) converges to the unconstrained global minimum first, and so avoids local minima and singularities on the constraint surface. This convergence is efficient, because the fitting error $f(\mathbf{x})$ is well approximated by a quadratic, and so the Hessian $\nabla^2 f$ is almost constant. When the Hessian $\nabla^2 f$ is constant, the best-fit surface is a linear subspace with zero curvature and no singularity, a lot simpler than the constraint surface.

The convex fitting function $f(\mathbf{x})$ can be viewed as a regularizer in the solution of $\mathbf{dx}$. It makes the constraint problem well-posed by using empirical data whenever additional constraints are needed to pin down free variables in $\mathbf{h}(\mathbf{x}) = \mathbf{0}$. Levenberg and Marquardt have shown that varying $\sqrt{c}$ by factors of 10 is an effective method to force convergence for nonlinear systems [Marquardt, 1963, Press *et al.*, 1988].

3 Implementation

This algorithm was implemented in C++ as an object recognition sub-system of an image analyst's workstation being developed by General Electric. The workstation provides facilities for displaying images, overlaying line drawings and annotations, performing geometric manipulations (scaling, panning, and control point warping, etc.) and low-level image processing functions (convolution, contrast adjustment, edge detection, etc.).

To use the object recognition system, the operator selects a model from the library and selects the region of interest in the image. The system segments that region of the image if necessary (it may have it cached from a previous run), then attempts to find a subset of the vertices in the region that are consistent with a projection of the model, using the techniques described in the previous sections. A confidence value, or response, is calculated as the value of the objective function for the final configuration as the percentage of the maximum attainable value of the objective function. The final configuration of the template and the confidence value are displayed. The flow of the data through the system is shown in Figure 1.

The following sections describe this process in more detail.

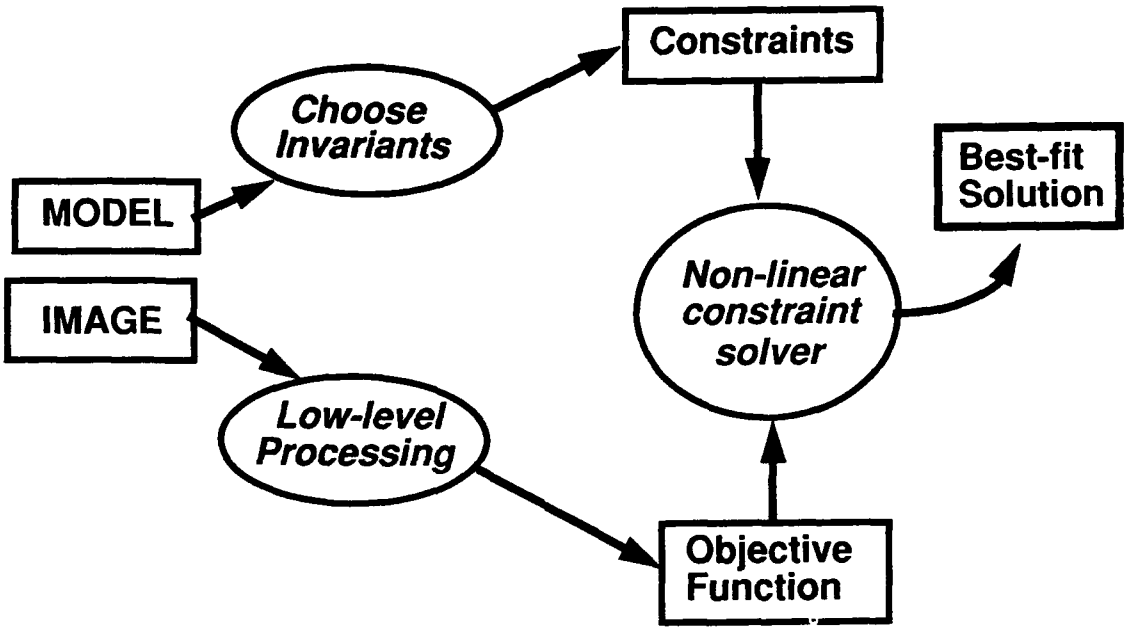


Figure 1: The flow of data through the SAR image analysis system.

3.1 *Area Selection and Image Segmentation*

The user starts the processing by selecting an area of the image with the mouse. The user places a rubber-banding rectangle around the area in the image under consideration. With the area selected, a right-mouse click starts the processing.

The image is segmented, if necessary (it may be cached from a previous run) into edges and vertices using a modified form of the Canny edge detector followed by line segmentation based on chain curvature extrema [Canny, 1983, Asada and Brady, 1984]. The program next selects all the vertices from the segmentation of the image that are contained within the region of interest created by the user.

3.2 Determining the Initial Configuration

There is likely to be quite a few vertices contained within the region of interest. The model, however, relates exactly five vertices in the invariant relationship. This creates the problem of determining the starting configuration for the template features. Upon initial consideration, the number of possible initial configurations appears to make the problem nearly intractable, however a number of simple and fast filters can be applied which reduce the number of candidates considerably.

1. The initial configuration does not need to be very accurate. This allows groups of adjacent vertices to be merged and replaced by their centroid.
2. The positions of the vertices in the initial configuration must approximately satisfy the projective invariants defined by the model.
3. The topology of the model is known, therefore assignments which would result in an impossible configuration can be ruled out.

By using these three strategies, we are typically able to reduce the possible initial configurations by a factor of over 10^5.

3.3 Running the Constraint Solver

At this point there exist a number of possible initial configurations (in our experiments as many as 50) that could potentially lead to the correct solution. For each configuration, a separate instance of the constraint solver process is spawned. Since each instance is independent, this allows for parallel processing by spawning shells to various available machines. As each instance of the constraint solver finishes, it outputs the final configuration and the value of the objective function. Higher values of the objective function indicate greater agreement between template (and therefore model) features and image features. All the resultant values are sorted, the largest is selected and the corresponding assignments are displayed. A response figure is calculated as a percentage of the maximum attainable value of the objective function.

3.4 Map Overlay and Warping

Now the correspondence between the model vertices and the image vertices which produce the greatest response is known. If the response is greater than a prespecified level (for example, 85%), this determines that the user did in fact select an area of the image that corresponds

to the model. Then using the model/image vertices correspondence, it is a simple matter to warp the map and overlay it on top of the SAR image. This not only positions and orients the instance of the selected model for the user, but also many other features on the image that are not easily seen.

4 Description of the Experiment

A sample image used in this experiment is shown in Figure 2. The image encompasses an area of about 20 miles by 20 miles in northwestern Pennsylvania. The most prominent features are 3 small lakes. Some other easily visible features include roads and cleared powerline corridors.

The United States Geological Survey (USGS) map of the same area was located and digitized. A section of this map is shown in Figure 3.

The largest of the three lakes (Twelve Mile Pond) was selected as the image feature we wished to locate. A model of this lake was constructed using the five-point invariant and data measured from the map. Five points around the perimeter of the lake were selected for the model. The points were chosen by selecting points of the high curvature around the lake. Contours of high curvature are most likely to be detected no matter what the viewing angle of the image.

Figure 4 shows the result when the region of interest contains an instance of the modeled lake. The response is 99%. This means that after the constraint solver adjusted the template, nearly perfect agreement of model and image features was obtained.

In Figure 5, we have performed a control point warp on the map, using the resultant assignments of model and image features, and overlayed it on the original image.

Figure 6 show a negative result. The region of interest does not contain an instance of the modeled lake and consequently the response is 9%.

The total execution time for all of these examples was less than two minutes on a single Sun SPARCstation 2. The majority of the execution time is consumed by the multiple constraint solvers. When the system is allowed to run constraint solvers on multiple machines the running time drops in proportion to the number of machines available.

5 Conclusions and Future Work

In this paper we have described a technique for interpreting images produced by Synthetic Aperture Radar that draws upon work in the areas of projective invariants and deformable templates and have shown some sample results of experiments with this technique. Work is currently underway to expand the system to use a variety of projective invariants, to allow the use of edge and area data from images, as well as non-planar models.

Acknowledgments

The authors would like to thank Andy Ochadlick of the Naval Air Development Center for providing the SAR image used in this paper.

References

[Asada and Brady, 1984] H. Asada and M. Brady. The curvature primal sketch. In *Proc. IEEE Workshop on Computer Vision: Representation and Control*, 1984.

[Barrett *et al.*, 1991] E. B. Barrett, P. M. Payton, N. N. Haag, and M. H. Brill. General methods for determining projective invariants in imagery. *Computer Vision, Graphics, and Image Processing*, 53(1):46–65, Jan. 1991.

[Brill and Barrett, 1983] M. H. Brill and E. B. Barrett. Closed-form extension of the anharmonic ratio to n-space. *Computer Vision, Graphics, Image Processing*, 23:92–98, 1983.

[Canny, 1983] J. Canny. Finding edges and lines in images. Report AI-TR-720, MIT, Artificial Intelligence Laboratory, 1983.

[Coelho *et al.*, 1991] Chris Coelho, Aaron J. Heller, Joseph L. Mundy, David Forsyth, and Andrew Zisserman. An experimental evaluation of projective invariants,. In *Proc. First Joint DARPA-ESPRIT Workshop on Applications of Invariance in Computer Vision*, 1991.

[Forsyth *et al.*,] David Forsyth, Joseph L. Mundy, Andrew Zisserman, Chris Coelho, Aaron Heller, and Charles Rothwell. Invariant descriptors for 3d object recognition and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. To appear.

[Lipson *et al.*, 1990] P. Lipson, A.L. Yuille, D. O'Keeffe, J. Cavanaugh, J. Taaffe, and D. Rosenthal. Deformable Templates for Feature Extraction from Medical Images. In *Proceedings of the European Conference on Computer Vision*, pages 413–417, Antibes, France, April 1990.

[Marquardt, 1963] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society of Industrial Applied Mathematics*, 11(2):431–441, 1963.

[Moffitt and Mikhail, 1980] F. H. Moffitt and E. Mikhail. *Photogrammetry*. Harper & Row, New York, third edition, 1980.

[Mohr and Morin, 1990] Roger Mohr and Luce Morin. Geometric solutions to some 3d vision problems. 1990.

[Press *et al.*, 1988] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1988.

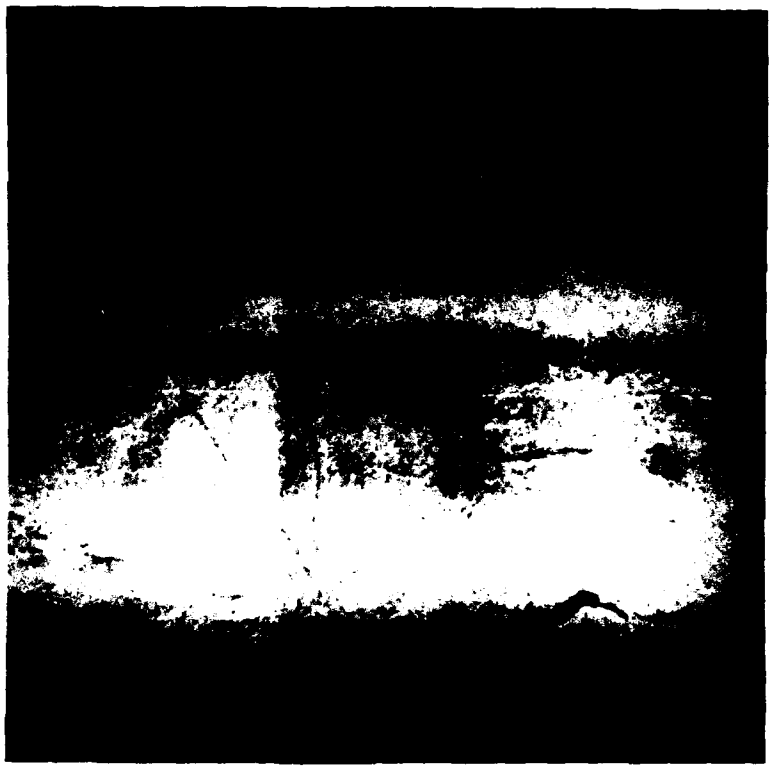


Figure 2: An oblique SAR image showing an area about 400 square miles in northwestern Pennsylvania.

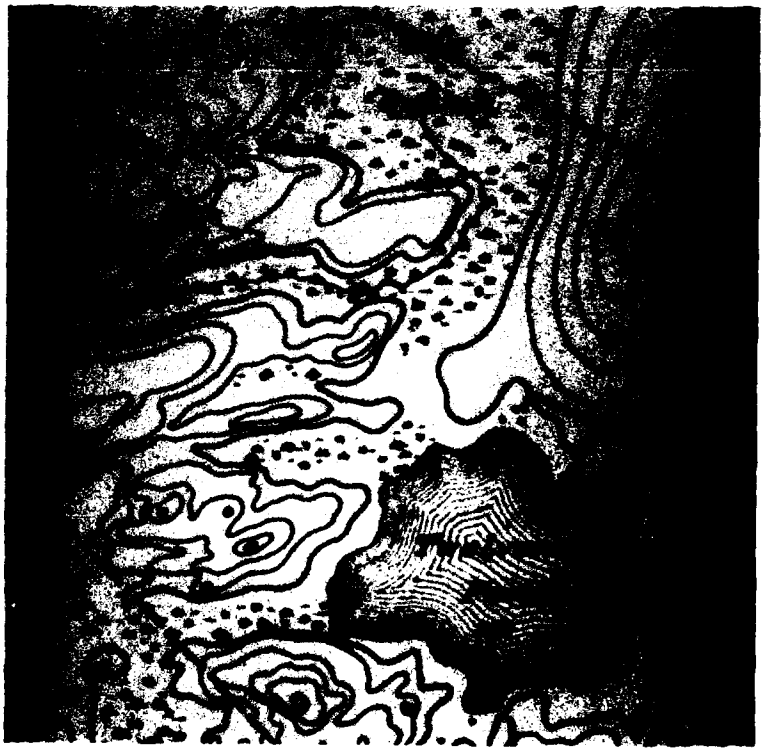


Figure 3: A portion of the USGS map of the same area.

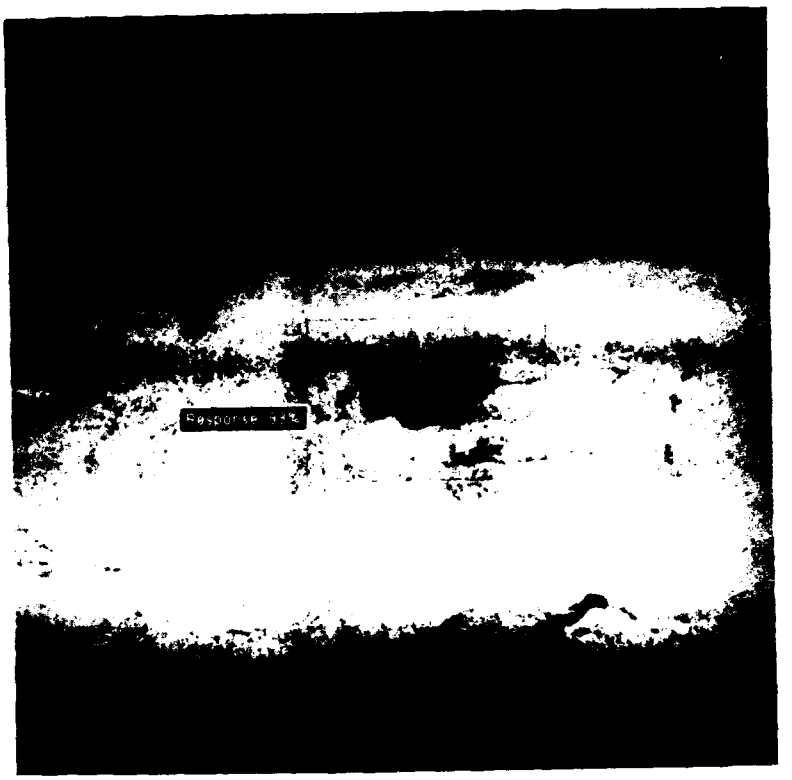


Figure 4: The result when the region of interest contains the modeled lake.

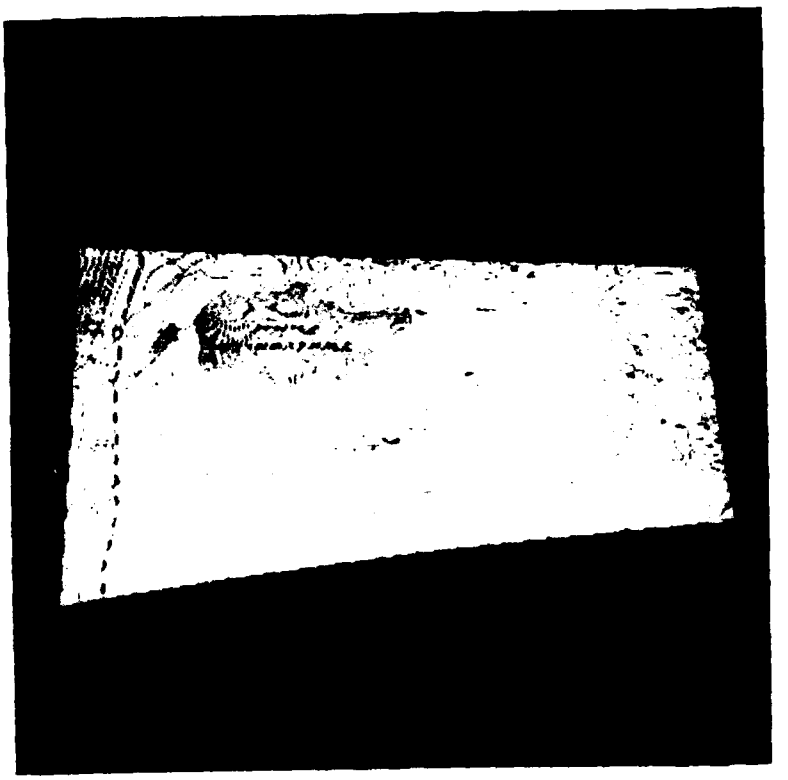


Figure 5: The original image with the map warped and overlayed.

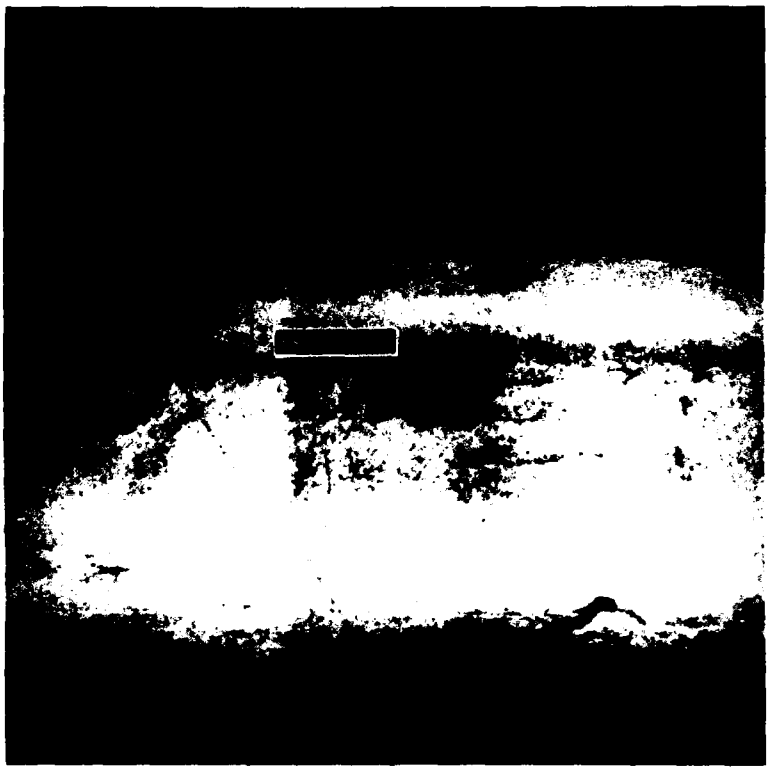


Figure 6: a sample result when the region of interest does not contain the modeled lake.

Refinement of Disparity Estimates Through the Fusion of Monocular Image Segmentations

David M. McKeown, Jr. and Frederic P. Perlant

Digital Mapping Laboratory
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

It is generally believed that the detailed analysis of remotely sensed imagery requires the extraction of a variety of partial image domain cues coupled with the use of *a priori* or contextual information. In some cases there are fundamental limits to the variety and type of information that may be extracted from a single image or stereo pair. However, in most cases a sufficient variety of cues can be extracted; the major issue is in how to utilize disparate scene cues to achieve a more complete and accurate overall scene interpretation.

The focus of this paper is to examine how estimates of three-dimensional scene structure, as encoded in a scene disparity map, can be improved by the analysis of the original monocular imagery. This paper describes the utilization of surface illumination information provided by the segmentation of the monocular image into fine surface patches of nearly homogeneous intensity to remove mismatches generated during stereo matching. These patches are used to guide a statistical analysis of the disparity map based on the assumption that such patches correspond closely with physical surfaces in the scene. We present refinement results on complex urban scenes containing various man-made and natural features and demonstrate the improvements due to monocular fusion with a set of different region-based image segmentations.[1]

[1]This research was primarily sponsored by the U.S. Army Topographic Engineering Center under Contracts DACA72-87-C-0001 and DACA76-91-C-0014 and partially supported by the Avionics Lab, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Topographic Engineering Center, or the Defense Advanced Research Projects Agency, or of the United States Government.

1. Introduction

One common problem for systems that interpret multiple sources of sensed data is the fusion of partial results from a variety of sources. This problem appears under many guises. For example, given a set of different scene descriptions generated from a single image using a variety of image analysis techniques, how does one intelligently combine such partial information? [Shufelt and McKeown 90]. The introduction of additional sensor types, temporal imagery, and multiple-look imagery create dimensions along which information fusion must be performed; as such, the complexity of the problem can increase. In some cases, increased amounts of data provide improved information. This may not necessarily follow, however; complex systems having different sources of error may not reinforce correct partial interpretations nor refute incorrect ones.

Thus, the key issue is the integration of many different sources of partial information. In computer vision (and in particular, three-dimensional scene analysis), the goal is to generate an *interpretation* of the scene that is as close as possible to the actual scene imaged. Such an interpretation can include the delineations and heights of buildings, a digital elevation model, and the centerline and width of roads in a transportation network. Our belief is that no individual computer vision technique can reliably provide a complete scene reconstruction. To achieve good performance, we need to gather a variety of information, extracted by various processes from the imagery, and synthesize this disparate information into a consistent model. Figure 1 shows a possible structure for such a scene interpretation system.

From the three-dimensional scene (G) we generally acquire two-dimensional imagery generated by a variety of different sensors. For example, a stereo pair of intensity images would represent such an imagery. As is well understood, the problem of interpreting the two-dimensional image (I) as a three-dimensional scene is underconstrained. In certain cases, we may have access to high-level knowledge about the contents

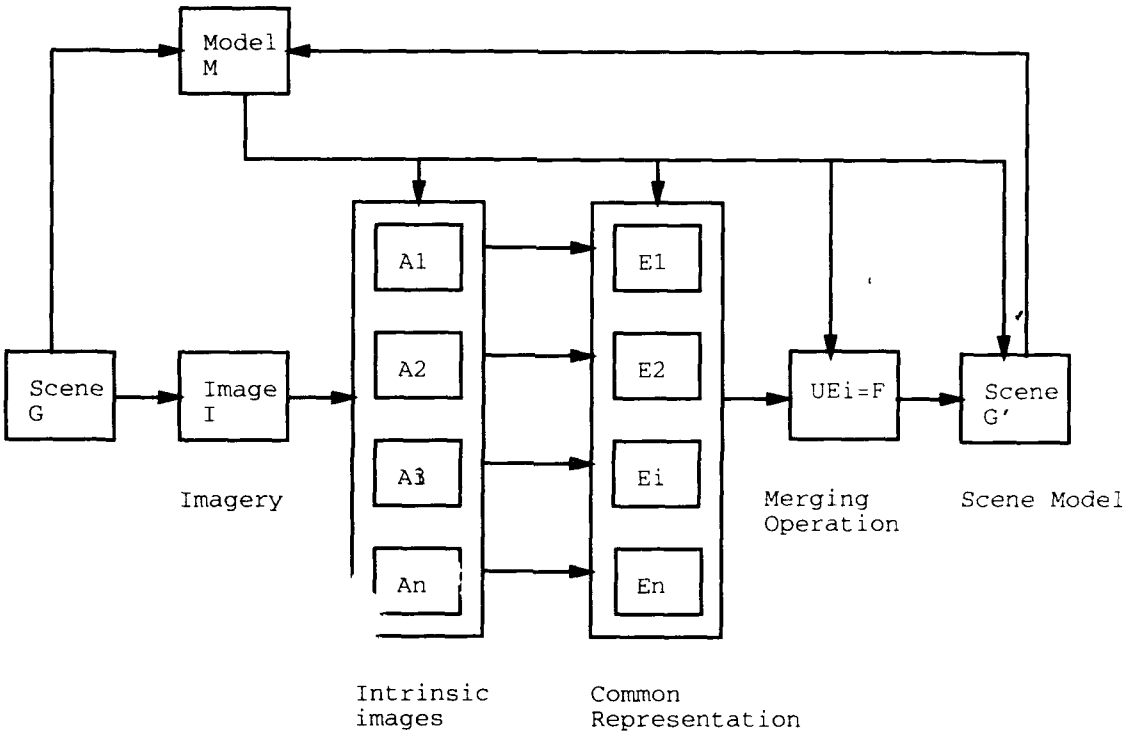


Figure 1: Data fusion in image analysis

of the scene, or particular objects that can be found in the scene. Such knowledge can loosely be called a Model (M). For example, in the case of aerial imagery we may have knowledge about the sensor resolution, the general characteristics of the scene (airport, urban area, rural area), etc. From the representation (I), we try to extract features that will allow us to interpret the scene {Ai}. These features are typically segmentations, edge maps, disparity maps, intensity maps, and the like. These can be thought of as a set of intrinsic images and primitives for intermediate and high-level vision [Barrow and Tenenbaum 78, Poggio 88]. In order to fuse the information embodied in these different "images", we need a common framework of representations (formed by the {Ei}). This framework needs to allow many, if not all, of the {Ai} features to be represented. The utilization of a common representation makes information fusion simpler and allows the generation of an interpretation (F), which then allows the generation of our scene model (G'). This model can be used to iterate through the fusion process again in conjunction with extra knowledge about the scene obtained from (M). This initial interpretation of the scene can help in the extraction of features {Ai}, the transformation of the features in the common representation, the merging process, and even the generation of the scene model.

Depending on the interpretation of the scene for which we are looking, we may need a varying amount of information; in most cases, more information is generally desirable. For instance, many techniques extract most of the necessary information for scene interpretation from a single intensity image; such techniques are said to apply *monocular analysis*. It is possible to take advantage of stereo disparity, however, to obtain more information that may be useful for disambiguation of monocular interpretations. Techniques utilizing stereo imagery are said to apply *binocular analysis* or *stereo analysis*. Other information such as global constraints or world models can be useful for further interpretation and disambiguation, but we believe that stereo analysis is a necessary step towards a coherent interpretation of the scene.

In this paper we describe a technique to merge information extracted from aerial imagery using a common region-based representation and show how disparate scene cues can be integrated to achieve a more complete and accurate overall scene interpretation. The particular task at hand is the refinement of stereo disparity data using monocular intensity information. In Section 2 we describe techniques to improve the accuracy of a stereo disparity map using a single segmentation of the left intensity image of a stereo pair. Thus, we are able to recover from mismatches generated during stereo matching by re-utilizing the intensity image that was originally used in the matching process. In Section 3 we discuss some experimental results on disparity refinement and describe techniques that allow for the integration of additional scene segmentations to provide for a more robust refinement process. Finally,

in Section 5 we give some future directions of this work in building extraction and built-up area analysis and speculate on how these techniques could be integrated into a more general three-dimensional scene interpretation system.

2. One approach to information fusion

In our research we utilize scene domain cues derived from monocular analysis and stereo analysis of left/right stereo image pairs. In the case of monocular analysis, one source of information is a region based segmentation of the left or right image. In the case of stereo analysis, our cues are primarily disparity maps derived from area-based (S1) and feature-based (S2) stereo matching systems [McKeown 86, Hsieh, et. al. 92].

These image-based cues provide different information concerning properties of man-made structures and terrain surfaces in the scene. In the case of three-dimensional reconstruction, we can make the assumption that the scene is composed of surfaces whose information content is primarily in terms of surface orientation and radiometry. Under these assumptions, we will see how estimates of three-dimensional scene structure (as encoded in a scene disparity map) can be improved by the analysis of the original monocular imagery.

Previous work in the general area of surface fitting include the use of planar patches [Eastman and Waxman 87] and both planar and quadratic patches [Hoff & Ahuja 89] to fit sparse stereo data to object surfaces. Work in the area of analysis and correction of stereo matching as a post-processing step include [Mohan & Nevatia 89] and [Cochran & Medioni 89]. The work of Luo and Maitre [Luo 90] used a single image segmentation composed of relatively small patches as a basis for fitting planar and quadratic surfaces. We use this basic idea, but simplify the patch fitting to one based solely on a statistical estimate of the disparity values. We extend this idea to include multiple segmentations, each providing a different estimate of the location of the partial object surface.

We have two sources of information that can be viewed as different representations of the physical surfaces found in the scene: disparity maps resulting from different stereo matchers providing the heights of the surfaces in the scene and the initial intensity images representing the radiometric properties of the surfaces in the scene. Figures 2, 4, and 5 are examples of "initial" data used for these data fusion experiments. Figure 2 is a high resolution aerial image containing a variety of buildings with complex shapes, typical of an industrial area. Figures 4 and 5 are disparity maps computed using S1 area-based and S2 feature-based methods, respectively. Figure 3 is a reference ground truth disparity map derived using a manual 3 dimensional segmentation of the scene. These images are a few of the many possible intrinsic images, {Ai}, in our general framework. It is important to note that, as in the intrinsic image paradigm, these two sources of information are "registered". That is, there is a pixel-by-pixel correspondence between points in the intensity image and points in the disparity map. In some many cases one issue complicating the use of multi-source information is the accurate registration or correspondence between the information sources themselves.

An intensity image, subject to sampling and digitization errors, poses difficulties for monocular analysis techniques such as segmentation. On the other hand, most stereo matching algorithms are fooled by different variations in the stereo pairs, which cause mismatches in the disparity maps. The mismatches in disparity maps primarily result from geometric and radiometric differences in the left and right images, rather than local digitization or sampling errors in the intensity images. Thus, it is possible to use information from the intensity images to reduce the number of mismatches introduced by stereo matching processes.

2.1. Region based interpretation

Our approach utilizes surface illumination information, provided by the segmentation of the monocular images into fine surface patches of nearly homogeneous intensity, to remove mismatches generated during stereo matching. First, we segment the intensity image into uniform intensity regions. These regions correspond to approximately planar surfaces in the image. We assume that the orientation and surface material are the primary factors for the radiometry of the image. Under these assumptions, uniform image radiometry is produced by a planar surface, of a certain orientation and material, in the scene.

These surfaces should have continuous linear disparity values (i.e., the disparity values of these regions are represented by continuous linear functions). Since the disparity map contains some noise, however, most of the regions segmented in the intensity image have disparity functions that are neither linear nor continuous. Ideally, we would like to approximate the actual disparity functions over the uniform intensity regions by the appropriate linear functions.

The problem of approximating a surface in three-dimensional space to a reasonable planar surface is a difficult one; we approximate such surfaces by horizontal surfaces. Then, the disparity values for each region will be the same for each pixel, and the problem is reduced to the selection of the best value for the heights of these surfaces. The general problem is that of locating of the surface which satisfies the equation

$$ax+by+cz+d=0$$

Given (x,y), we should be able to obtain

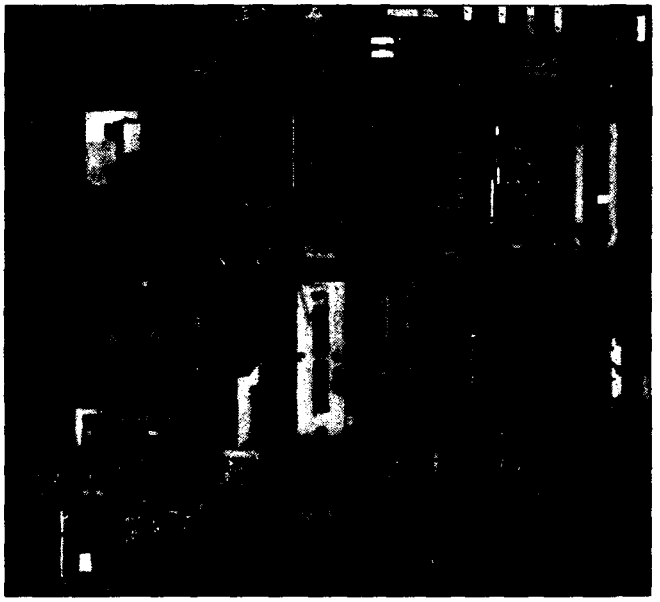


Figure 2: DC38008 Industrial Scene



Figure 3: DC38008 Disparity Reference

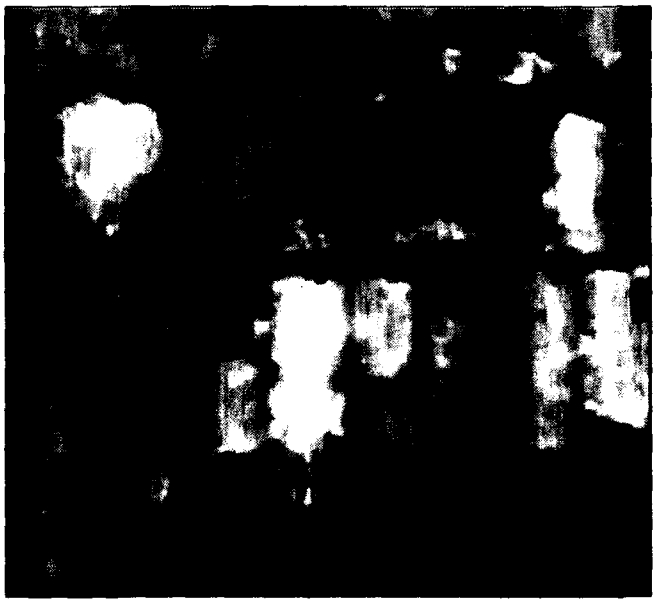


Figure 4: DC38008 S1 Disparity



Figure 5: DC38008 S2 Disparity

$$z = (-ax-by-d)/c$$

We assume here that z'= -d'/c' only. Then the problem is to find (-d'/c') that best fits the surface so that

$$ax+by+c*(-d'/c')+d\sim=0$$

or to find z' so that z-z' would have a minimal value over the region (this can be the weighted mean of the z distribution or the most 'representative' value of the z distribution). In other words, we need only select a single disparity value for each region. Since we are using an over-segmentation of the image, a piecewise planar disparity map gives a good approximation of the relief in the scene. Furthermore, since we are interested in building extraction in aerial images, this approximation will be adequate.

This region-based interpretation has been developed for two different applications. We show how this approach can support information fusion from different segmentations and well as across multiple disparity estimates based upon a local decision making evaluation. In Section 3 we describe how improved disparity maps may be obtained by correcting the mismatches produced by stereo matchers and by refining the disparity discontinuities.

2.2. Intensity Segmentation Techniques

The general scene segmentation problem is, of course, a very difficult one and has a long history in image processing and computer vision. There are no universal segmentation techniques that work well

across a variety of imagery and tasks. Such low level algorithms typically differ in their approaches; they may utilize intensity-based, area-based, or edge-based techniques. Some systems combine these techniques into hybrid algorithms. We have concentrated on those segmentation methods that produce (nearly) uniform intensity regions because we wish to detect those image regions that correspond to oriented surface patches in the scene. We utilize a region segmentation algorithm based upon the histogram splitting paradigm [Ohlander 78] and a region growing algorithm [Yakimovsky 76] which takes into account edge strength and shape criteria [McKeown 84]. Interestingly, while neither of these methods give completely satisfactory segmentation results, they provide good over-segmentations that rarely merge object/background boundaries. Both techniques will also provide different segmentations based upon modification of a small set of parameters.

In our initial experiments we generated three scene segmentations; two by using different parameters for histogram selection, and one by using region growing. These segmentations provided the basis for our work in intensity/disparity fusion, the goal of which was to produce an improved three-dimensional scene interpretation. Figures 7, 9, and 9 show examples of these segmentations on the DC38008 industrial intensity image. As a preprocessing step, we smoothed the original image (Figure 6) using a Nagao filter.

2.2.1. Machineseg

One of the major difficulties with region growing techniques in complex scenes is the difficulty in determining automatic stopping conditions for the merging procedure. MACHINESEG [McKeown 84] is a region growing system that tries to preserve edges between regions and stops the growing procedure when certain shape or spectral criteria are not satisfied inside the region. It adds a decision procedure to evaluate the effect of the next merge operation and either allows the merge to proceed or to be rejected. In the case of disparity map refinement, we want the regions to be sufficiently uniform that they could be treated as planar (or at least "soft") surfaces. We also limited the size of the generated regions so that very small regions could not be generated, as these could be considered noise or non-representative regions. Since we are not considering the regions smaller than our noise threshold (20 pixels) our segmentation in Figure 7 is not a complete partition of the image. However, it does produce regions for most of the important surfaces in the image.

2.2.2. Colorseg

This histogram splitting technique is based on the extraction of regions with limited intensity ranges (in other words, region of approximately uniform intensities). The technique searches for the peaks in the histogram of the image and segments the regions whose intensity values fall in windows around these peaks. The regions are then removed from the image and the process continues until all the pixels in the image have been removed. This process results in a segmentation composed of connected regions, each having an intensity range less than a certain threshold. This technique does not guarantee preservation of the edges (in particular, small edges) but it may ignore local noise with strong edges that other techniques will classify as regions. As in the previous technique, we removed very small regions (less than 20 pixels) that could be considered as noise, for further processing.

In our experiments, we generated different segmentations with different segmentation techniques. For instance, using the colorseg technique we generated two segmentations of the images, one with "uniformity" defined as a maximum of 10 intensity levels inside the region (to tolerate sensor noise and allow for imperfect planar surfaces) and another with "uniformity" defined as a maximum of 20 intensity levels (to tolerate more noise). An estimation of the noise or the average intensity range for the surfaces in the image is a delicate problem, and the use of different segmentations to estimate the intensity range inside the regions does not necessarily increase the reliability of the process. It is thus important that we obtain different segmentations of the scene that are *not consistent*, such as those in Figures 8 and 9. The fusion of these data may overcome some of the inherent problems of a single segmentation since they provide different local evaluation contexts for disparity estimates in the scene. In the following sections we show how we can merge information using different intensity segmentations.

2.3. Disparity map results

Our initial height information for the industrial scene was derived using two different stereo matching algorithms. Given these sets of height information, which may or may not be reliable or unique, it becomes necessary to use a data fusion process in order to maximize the amount of useful information gained from these sets of height estimates.

We used 2 different matching techniques, one area-based (S1) and the other feature-based (S2). S1 uses the method of differences technique on neighborhoods of the image in hierarchical fashion [Lucas 84, McKeown 86]. S2 performs a hierarchical matching of epipolar intensity scanlines in the left and right image [Hsieh 90, Hsieh, et. al. 92]. The results of these stereo matching algorithms are different; S1 gives us a dense disparity map (i.e., a map containing a disparity value for each pixel in the image), while S2 gives us a sparse disparity map (i.e., a map containing a disparity value for those pixels corresponding to peaks or valleys in the intensity images). We interpolate the sparse S2 matches into a dense disparity map by step



Figure 6: Nagao filtered DC38008

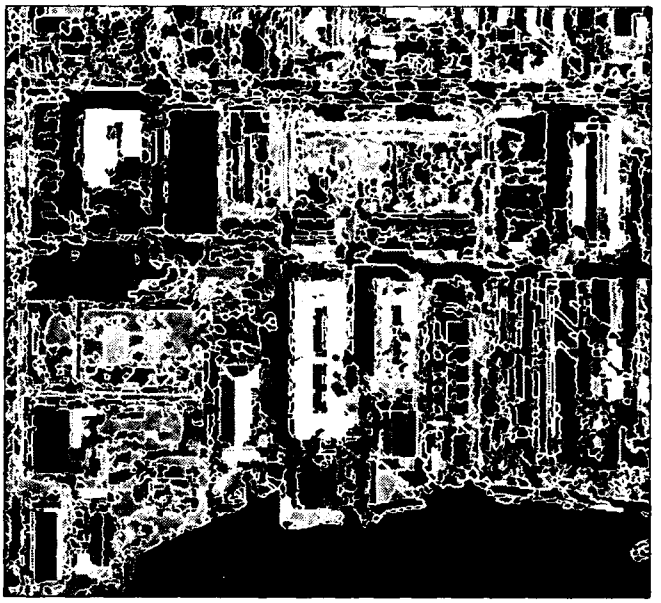


Figure 7: MACHINESEG segmentation

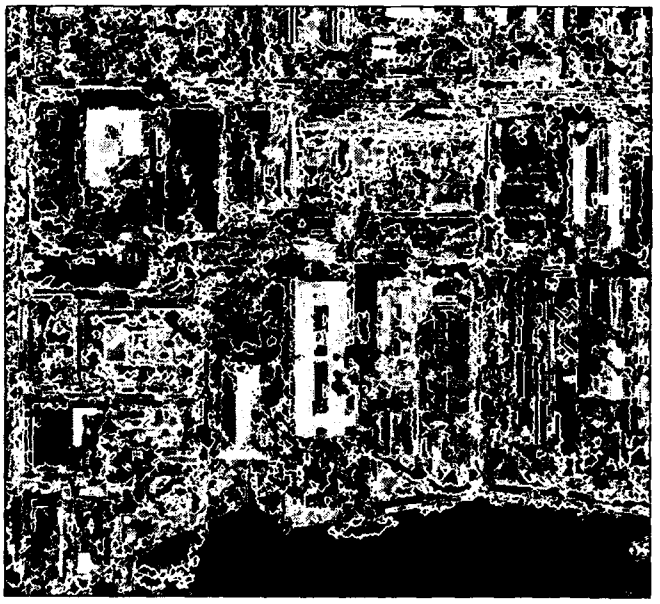


Figure 8: COLORSEG SEG10

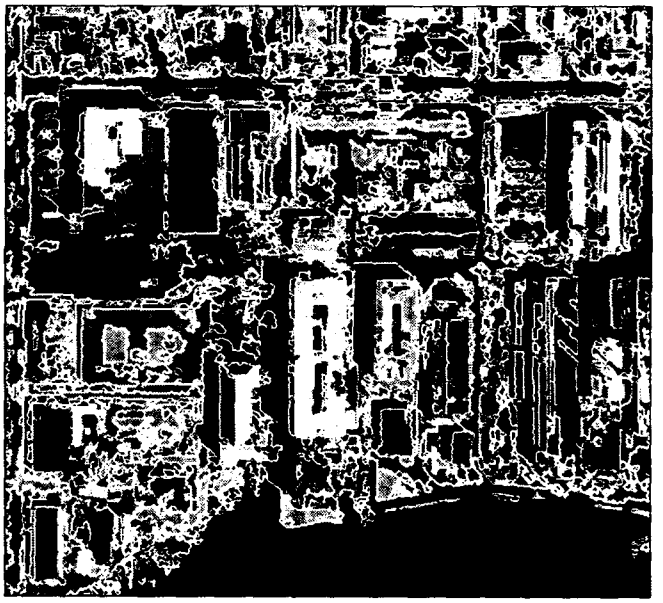


Figure 9: COLORSEG SEG20

interpolation. Our fusion mechanism will have to correct mismatches by the S1 and S2 stereo systems as well as those introduced by the interpolation method. One can easily observe differences in the disparity estimates produced by S1 and S2 shown in Figures 4 and 5, respectively.

3. Fusion Experiments

The goal of refinement is to remove mismatches, improve the location of disparity discontinuities, and to obtain the best height estimate for each point in the scene. The refinement process is can be decomposed into two stages. First, we perform monocular segmentation on the left image of the stereo pair. We generate a region representation containing intrinsic attributes of region area, intensity distribution, and adjacent neighbors. For each of the disparity maps we obtain a histogram of height estimates for each region in the segmentation. A set of disparity attributes is computed for each region in each segmentation including frequency statistics for the disparity values, and estimate of the 'best' disparity value, and a confidence score for this value. This allows the computation to proceed at a symbolic level on a region-by-region basis.

In the second stage we merge this information represented in the region data structure by selecting the 'correct' value from the available information, and by comparing scores based on the nature and quality of

the different pieces of information. The goal then is to evaluate the quality or confidence in the information so as to maximize the amount of accurate data we merge from our different information sources.

In general there are several approaches to disparity map refinement using monocular data. The combinations are quite obvious, and in the remainder of this section we describe experiments using the first three techniques:

1. Disparity refinement using one segmentation.
2. Disparity refinement using several segmentations.
3. Disparity refinement using one segmentation and several disparity.
4. Disparity refinement using several segmentations and several disparity maps.

3.0.1. Simple disparity refinement

In this first approach, a histogram is constructed for each segmentation region. The values of each histogram are the disparity values in each region. The most representative value of each histogram is then selected. In our case, this value was simply that of the highest peak in the histogram. We chose this value for two reasons. The step-interpolated S2 disparity maps result in disparity histograms having only a few values, which correspond to real height values or matching noise. If the matching is reasonably robust, the noise will introduce local maxima in the histogram that will be smaller in magnitude than the best height estimate. Further, a typical region histogram for an S2 disparity map exhibits one or two large peaks and a few noise peaks that influence the average value of the histogram, making it less reliable as a representative value.

For non-horizontal regions and S1 results, the average disparity may suffice for a reasonable measure of the height of the region. A confidence score can be generated for these disparity values based on the characteristics of the histograms (and, conceivably, on the type of disparity map used as well as the nature of the region histograms). Finally, this disparity value is assigned to the entire region, under the assumption that it will be a better estimate of the height for the whole region. In most cases, this removes a large number of the mismatches, but whenever our initial assumptions about scene radiometry are not valid, our height estimates may differ from the correct height value.

We implemented this experiment using COLORSEG at 10 (SEG10) and 20 (SEG20) intensity levels as previously shown in Figures 8 and 9. A refined disparity map based on these segmentation regions and the S1 and S2 disparity values was produced. Regions that did not generate a consistent disparity estimate, are set to black. Figures 10 and 11 show the results of the disparity improvement process for the SEG10 and SEG20 segmentations using the S1 disparity map. Figures 12 and 13 show the results of the disparity improvement process for the S2 disparity map.

A visual comparison between the original disparity maps and the refined maps is quite striking, particularly in the case of S1. The result of the refinement process seems to greatly improve the sharpness of the buildings, particularly using the coarser SEG20 segmentation. In the case of S2 the improvement is not as dramatic, but the building boundaries are sharpened and several of the building surfaces are made more homogeneous.

3.0.2. Multi-segmentation disparity refinement

In this second approach to refinement, we merge different height estimates, given different intensity segmentations(SEG10, SEG20) and then merge the results across the different segmentations. We refine the disparity estimate for each pixel by locating the intensity region to which it belongs, for each of the image segmentations. This list of regions can then be searched to obtain the disparity estimate attribute (computed for a given disparity map) as well as a confidence score for this estimate. The confidence score is then used to select the best disparity value, which is then assigned to the pixel. Currently a simple decision is made to select the disparity value having the highest confidence score.

An attempt is made to maximize the score for each pixel in the entire image. This is done by selecting a disparity value in all of the regions resulting from the union of the segmentations. In other words, the segmentations were merged and the best height value was selected for each of these regions, by utilizing the confidence scores computed for each region. The scoring method currently in use takes into account information about the nature of the segmentation used.

In particular, higher confidences can be assigned to sufficiently large regions in a constrained segmentation such as SEG10 than to the equivalent regions in SEG20. Information of this nature must be incorporated in the confidence function for each segmentation region.

Figures 15 and 14 show the results of merging the SEG10 and the SEG20 segmentations for the S2 and the S1 disparity maps, respectively. Depending on the confidence scores of the disparity values selected for each segmentation, we were able to obtain improved disparity estimates for some of the regions. Comparing these results to Figures 12 and 13, disparity maps obtained with the simple method, we observe some of the failings of both approaches. The initial segmentations, in some cases, are under-segmented instead of over-segmented, resulting in the grouping of regions that should have been assigned different height estimates. Another factor is the confidence evaluation function for the regions of the segmentation, which only takes simple properties of the disparity histograms of each region into account.



Figure 10: S1 refined SEG10



Figure 11: S1 refined SEG20



Figure 12: S2 refined SEG10



Figure 13: S2 refined SEG20

3.0.3. Multi-Disparity Disparity Refinement

In this approach, several different disparity maps are merged using a single segmentation, looking for consistent areas across disparity maps. This approach is similar to the simple disparity improvement approach, except that we now attempt to select the best disparity value based on a set of differing confidence scores. The score established for each disparity map at each pixel should be dependent on the stereo matching algorithm used to generate the map, and should also take into account the nature of the possible mismatches resulting from each stereo matching technique.

The major problem with all of the refinement approaches discussed in this paper is the development of a reasonable confidence evaluation function for each set of data. Currently, confidence is evaluated by a scoring function that utilizes the standard deviation and the disparity range of the histogram for each region, as well as the size of the region. Ideally, this scoring function would also take into account the nature of the disparity map. As an initial experiment, we defined a similar scoring function for each disparity map and checked for disparity consistency across segmentation regions. Figure 16 shows the result of this experiment using the MACHINESEG segmentation previously shown in Figure 7. Again, areas where disparity values differ significantly between S1 and S2 are marked in black. These areas often correspond to regions of occlusion



Figure 14: S1 refined with Merge SEG10 / SEG20



Figure 15: S2 refined with Merge SEG10 / SEG20

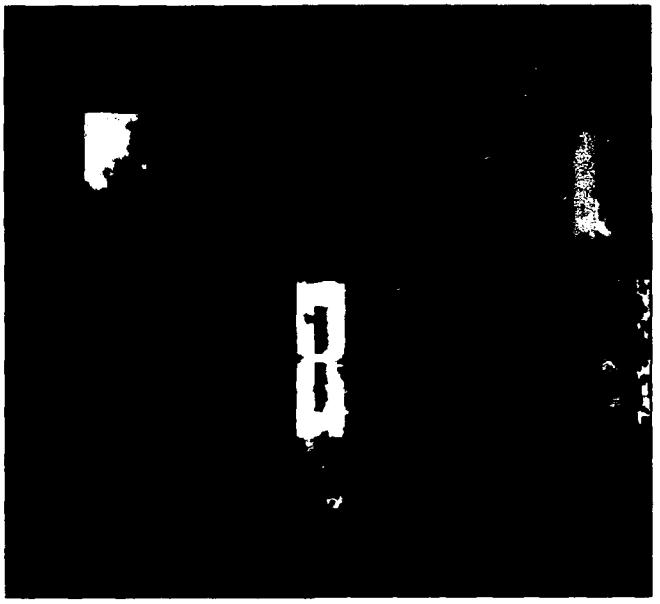


Figure 16: Merge S1 and S2 using MACHINESEG

and shadow. Future work needs to integrate other monocular cues, such as illumination and shadow analysis into this process [Irvin 89].

4. Performance Evaluation

In this section we give a quantitative performance evaluation of the improvement in disparity map accuracy when compared to a ground truth 3 dimensional segmentation of the scene. We also show results for two additional complex urban scenes, CIVIL1 and GAO1. Figures 17 and 18 show the left intensity image and reference disparity map for the CIVIL1 site. Figures 19 and 20 show the COLORSEG segmentations using 10 and 20 levels, respectively. Figures 21 and 22 show the original S1 and S2 disparity maps. Figures 23 and 24 show the result of multi-segmentation disparity refinement using SEG10 and SEG20. A parallel set of results for the GAO site is presented in Figures 25 to 32. The error analysis methodology has been described in detail in [Hsieh, et. al. 92].

4.1. Global Scene Error

Tables 1, 2, and 3 give detailed statistics for each of the

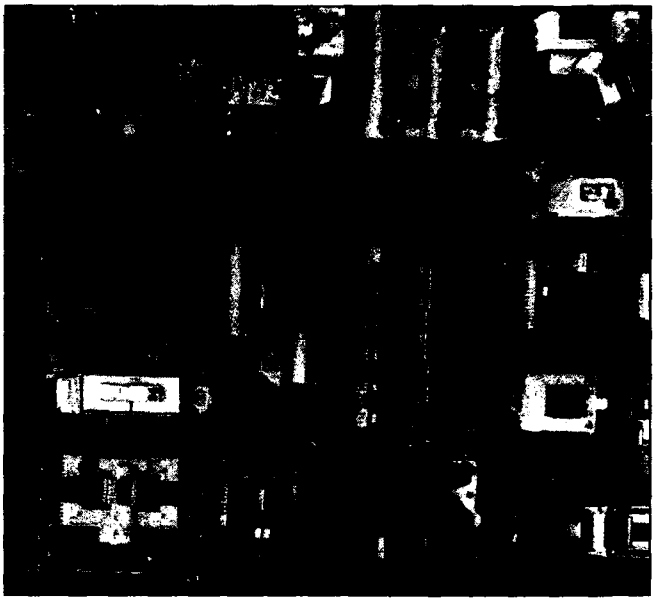


Figure 17: CIVIL1 Scene

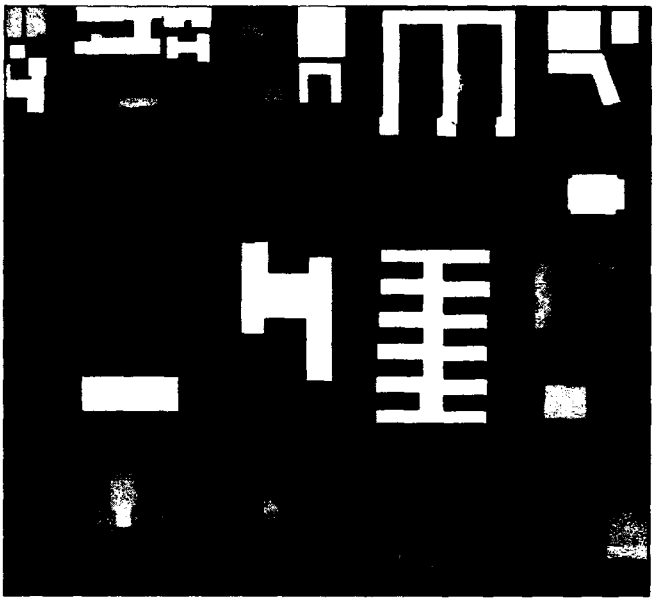


Figure 18: CIVIL1 Disparity Reference

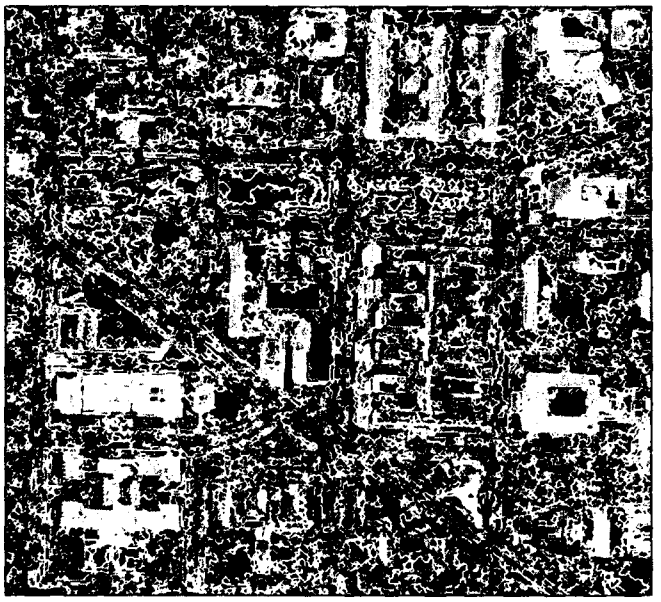


Figure 19: CIVIL1 COLORSEG 10

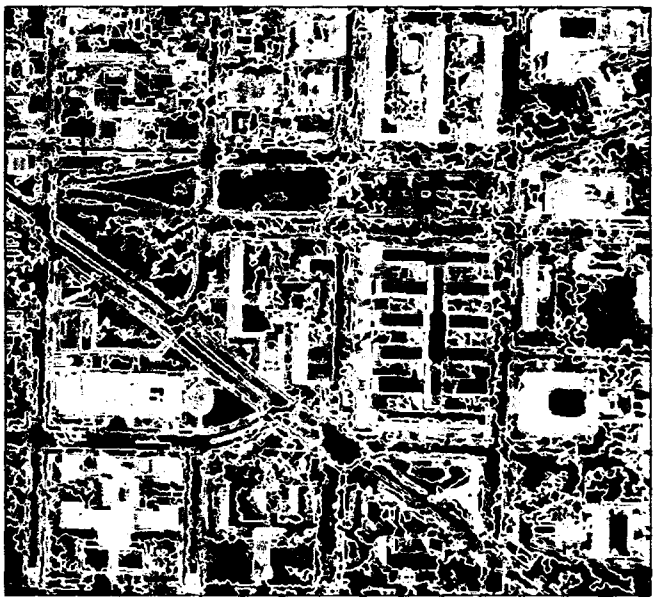


Figure 20: CIVIL1 COLORSEG 20

stereo matching systems, S1 and S2, and show the improvement due to our fusion technique. These statistics on based on the global error between the reference disparity map and the disparity results produce by the stereo systems and the refinement process. As a global measure of accuracy we present the average pixel disparity error for the entire scene and the percentage of points having an estimate within +/- one pixel disparity from the reference for the entire scene. The use of +/- one pixel disparity reflects some of the accuracy limitations in the reference disparity map. These simple parameters give us an idea of the magnitude of the errors in the scene, but do not give much insight into their distribution. Other error metrics such as min/max error are not very reliable since they can be caused by single point errors that may occur in either the calculated or reference disparity map.

It is clear from these tables that the refinement process quantitatively improved both the average error and the percentage of points within +/- one pixel of the reference disparity. Overall average pixel error on the three scenes ranged from 1.59 pixels for S2 on GAO1 down to .02 pixels for S1 refined in CIVIL1. The percentage improvement for refinement with respect to our within +/- one pixel disparity metric was consistently nearly 8-10% on all tests.

One interesting issue in interpreting the global statistics is knowing the frequency distribution of the disparities



Figure 21: CIVIL1 S1 Disparity



Figure 22: CIVIL1 S2 Disparity



Figure 23: CIVIL1 S1 Refined Disparity

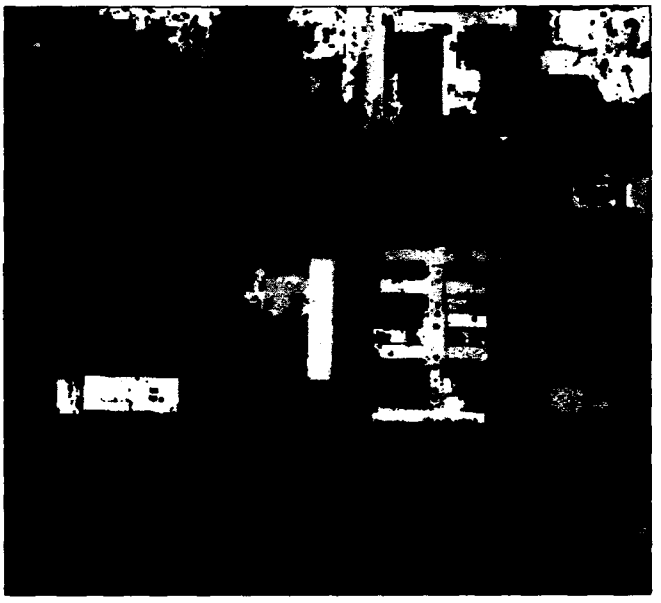


Figure 24: CIVIL1 S2 Refined Disparity

in the scene. Obviously error metrics based on small populations are not robust. Figure 33 plots the distribution of disparity values in the ground-truth reference disparity map for DC38008. One can see that the majority of the pixels are at, or close to, the ground. Figure 34 shows the distribution of disparity values for each of the stereo systems and their refinements. While the curves coincide quite well, there are noticable areas, particularly in the disparity range between 5 and 15 where none of the stereo results significantly approach the ground truth segmentation. Data for the other scenes, GAO1 and CIVIL1 exhibit a similar pattern. This distribution analysis is useful for understanding the structure of the scene, and in interpreting the meaning of the statistics presented in the following section.

4.2. Effects of Disparity Jumps and Structures

One way to address some of the issues that are hidden by the global statistics discussed in the previous section is to measure the influence of the actual disparity value on matching accuracy for each of the methods. That is, as we attempt to recover larger and larger disparity values, does our matching error increase? Alternatively, is the matching error independent of the disparity range? The graphics in Figures 35, 39, and 37 plot the average error in pixel disparity at each disparity level for each of the test examples. Each contains four graphs showing the results for S1, S2, and the refinement results for S1 and S2. In general, these graphs indicate that the greater the actual disparity, the more likely the various matching

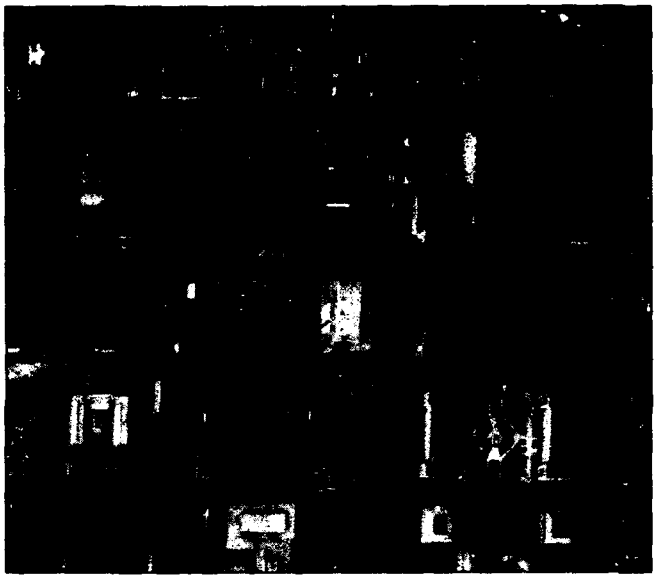


Figure 25: GAO1 Scene

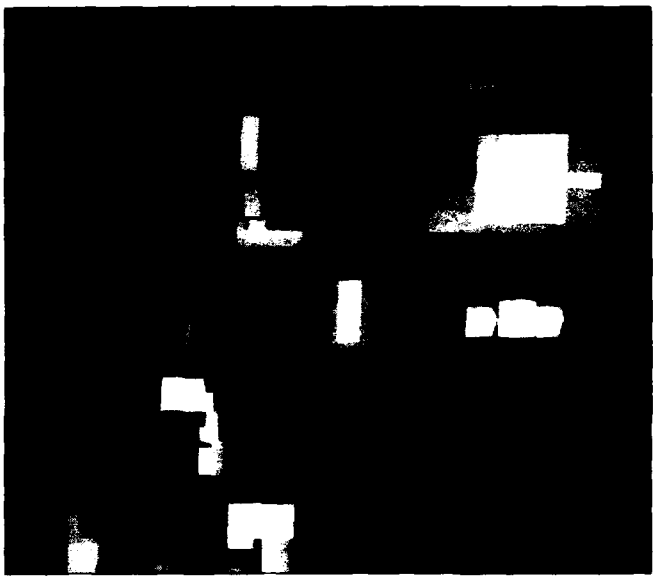


Figure 26: GAO1 Disparity Reference



Figure 27: GAO1 COLORSEG 10



Figure 28: GAO1 COLORSEG 20

algorithms will make a mistake. This is true for both larger positive and negative disparity. These errors are reflected in both a higher average error and a lower percentage of points within +/- one pixel of the actual disparity. However, as we observed in Figure 34 the number of pixels at higher disparity is significantly smaller, making the error estimates less robust that those for disparities between +/- 2 of zero.

In areas with man-made structures, global accuracy statistics such as those described in the previous section do not adequately convey the quality of the stereo matching system with respect to the buildings in the scene. In most cases, buildings may cover only a small portion of the scene, and the background terrain will statistically dominate the scene-wide estimate of disparity quality. Thus, we require a method that allows buildings to be evaluated independently or as a class of objects in the scene. Additionally, there are several metrics that can be used to evaluate both the disparity estimate and the quality of the depth jumps [Hsieh, et. al. 92]. For the purpose of this paper we show average disparity error in pixels for each of the buildings in the three scenes. We have assigned building ID's based upon a manually compiled ground-truth disparity map so that taller buildings have larger numeric ID's. Figures 36, 38, and 40 plot the average error in pixel disparity against regions IDs, sorted in increasing order of disparity. From these graphs one can get a sense that there are several cases where the refinement value for a particular building is worse than the original stereo estimate. Given the good global error improvements shown in Tables 1, 2, and 3, we are currently looking at the specific effect of



Figure 29: GAO1 S1 Disparity



Figure 30: GAO1 S2 Disparity



Figure 31: GAO1 S1 Refined Disparity



Figure 32: GAO1 S2 Refined Disparity

refinement in these buildings. Without such performance analysis tools it would be impossible to locate and understand these problems.

These statistics allow us to pinpoint problems at a much finer grain of detail than can be accomplished with global analysis. Thus we can identify specific buildings in the scene and try to understand, at the algorithmic level, whether there are specific situations where matching could be improved. Once identified, these improvements should have an overall positive effect on the rest of the scene. The result, of course, can be subjected to the same rigorous performance analysis. Once we commit to working on complex scenes, as opposed to synthetic controlled images, the visual inspection of disparity results to discover small variations in performance becomes very unsatisfactory, except possibly at the earliest stages of experimentation. Such manual inspection greatly limits our ability to detect subtle conceptual bugs or recognize possibilities for algorithmic improvement. We can perform systematic analysis across multiple scenes. For example, in applying statistics that take into account the disparity jump for individual buildings, we can aggregate performance information for all buildings across all scenes to achieve a larger statistical sample.

5. Conclusions

We have described a set of fusion processes that allow us to improve the quality of disparity maps, and we have demonstrated the use of information fusion to improve disparity map analysis. The major feature of

Global Error Estimate for Stereo Matching Using Figure 3 as ground truth				
Stereo Method	Min/Max Disparity	Average Error in pixels	% of points within +- 1 pixel disparity	Ground Truth Disparity Range
S1	-12/13	.11	61%	-2/14
S2	-5/14	.30	75%	-2/14
S1ref	-5/13	.08	72%	-2/14
S2ref	-4/14	.03	83%	-2/14

Table 1: Statistics for different stereo matching methods on DC38008

Global Error Estimate for Stereo Matching Using Figure 18 as ground truth				
Stereo Method	Min/Max Disparity	Average Error in pixels	% of points within +- 1 pixel disparity	Ground Truth Disparity Range
S1	-15/17	.16	50%	-9/20
S2	-9/20	.38	58%	-9/20
S1ref	-10/17	.02	59%	-9/20
S2ref	-7/20	.40	68%	-9/20

Table 2: Statistics for different stereo matching methods on CIVIL1

Global Error Estimate for Stereo Matching Using Figure 26 as ground truth				
Stereo Method	Min/Max Disparity	Average Error in pixels	% of points within +- 1 pixel disparity	Ground Truth Disparity Range
S1	-20/21	.16	35%	-2/32
S2	-1/27	1.59	45%	-2/32
S1ref	-20/21	.46	47%	-2/32
S2ref	-1/27	.96	55%	-2/32

Table 3: Statistics for different stereo matching methods on GAO1

the information fusion technique described here is the definition of a common frame for information fusion. The representation framework (an intensity segmentation) can be used in conjunction with

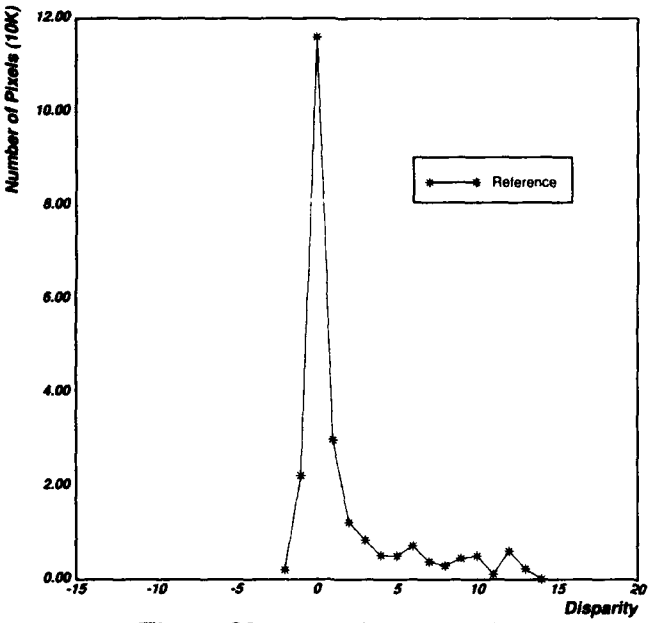


Figure 33: DC38: Ground-truth Disparity Frequency

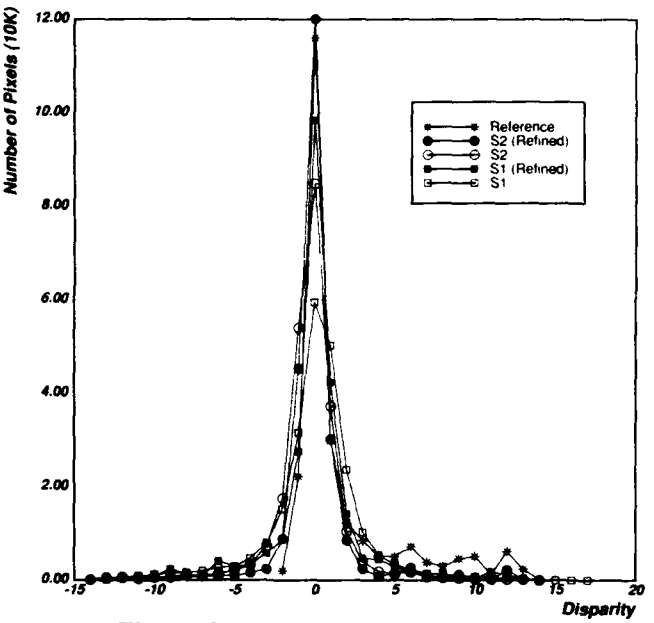


Figure 34: DC38: Stereo Matching Disparity Frequency

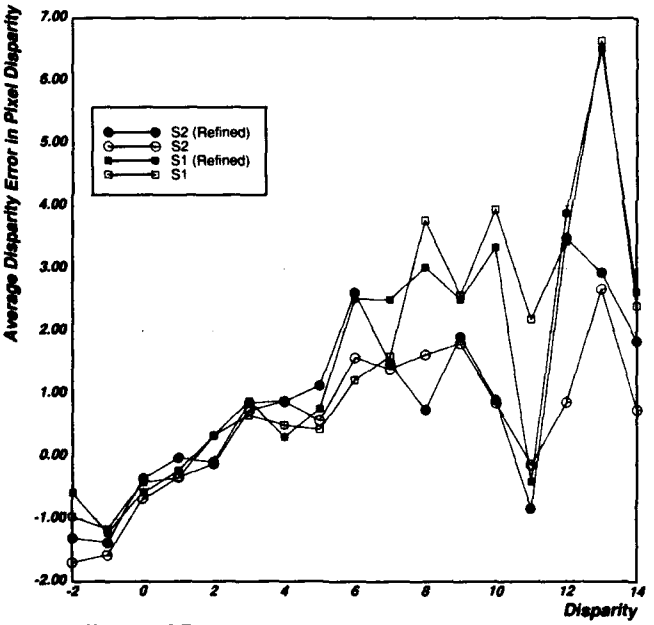


Figure 35: DC38: Average Disparity Error in Pixel Disparity

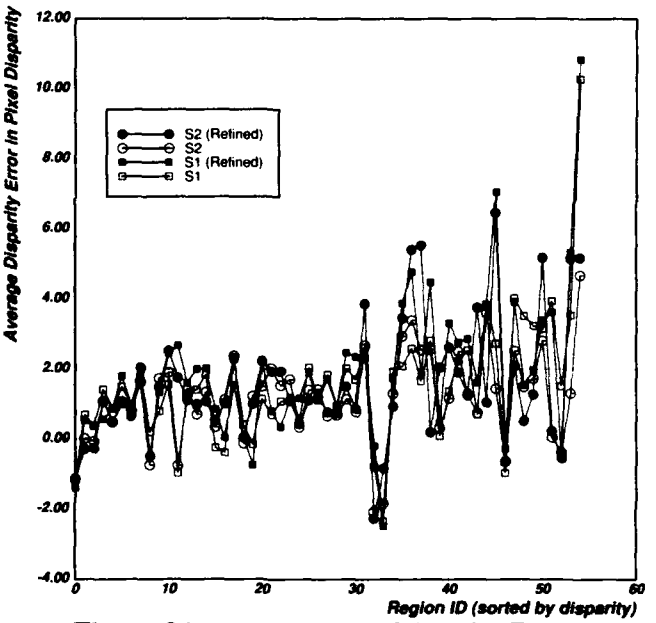


Figure 36: DC38: Average Disparity Error by Building

different types of intrinsic images. The approach developed here treats homogeneous intensity regions as surfaces, which allows three-dimensional information to be extracted readily.

Many research issues remain to be explored. The new disparity maps generated by the information fusion process contain regions which each have only one disparity value. In many cases, these unique values are not the best possible disparity estimates for the regions, and a refinement process may need to be invoked to correct these estimates. One approach might be to use the new disparity map itself as input to a verification process which could refine disparity estimates for each pixel or for those regions with low confidence scores.

Other sources of information could be utilized at the refinement stage to further enhance the disparity map. One promising approach would be the use of left/right consistency, such as left/right matching of low confidence regions or local correlation for these regions. Again, it would be important to use as much information as possible, while conservatively adjusting or refining data based on its confidence scores. In the

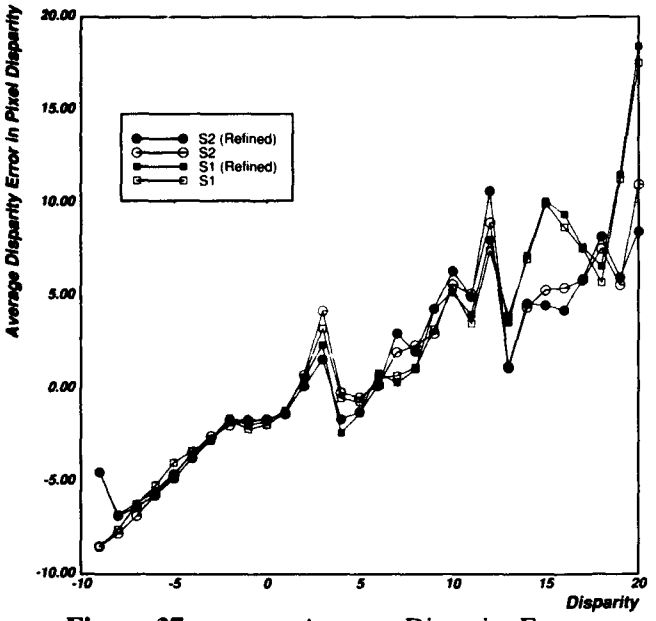


Figure 37: CIVIL1: Average Disparity Error in Pixel Disparity

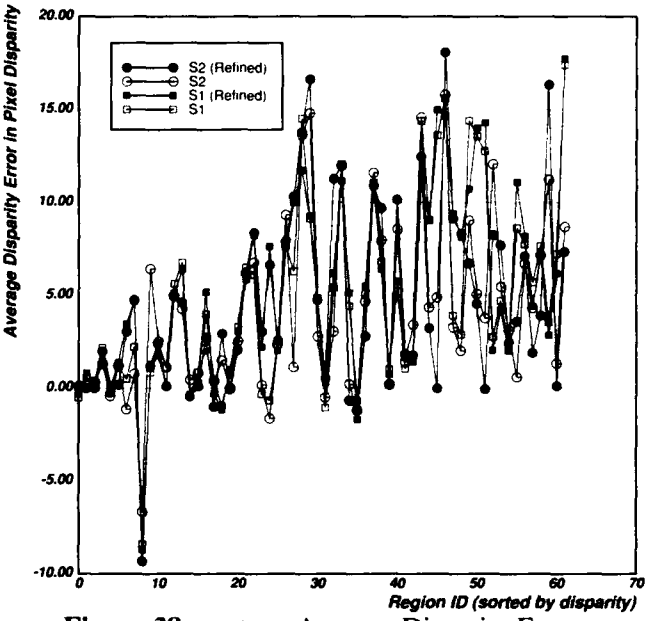


Figure 38: CIVIL1: Average Disparity Error by Building

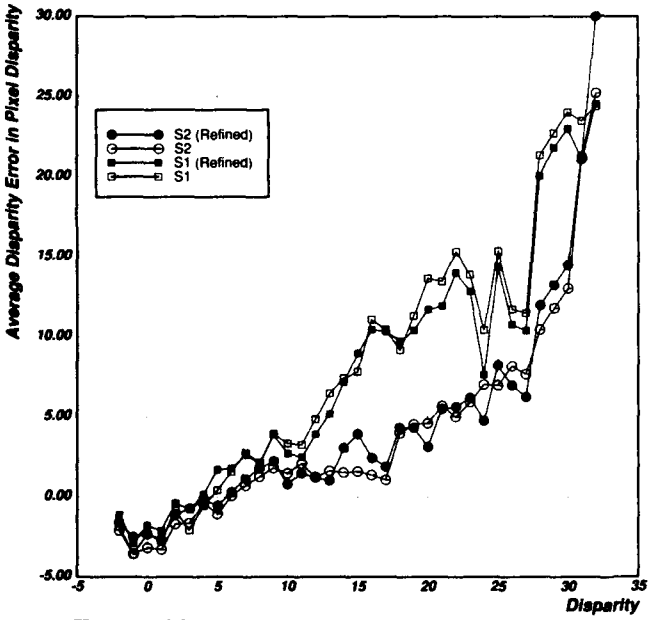


Figure 39: GAO1: Average Disparity Error in Pixel Disparity

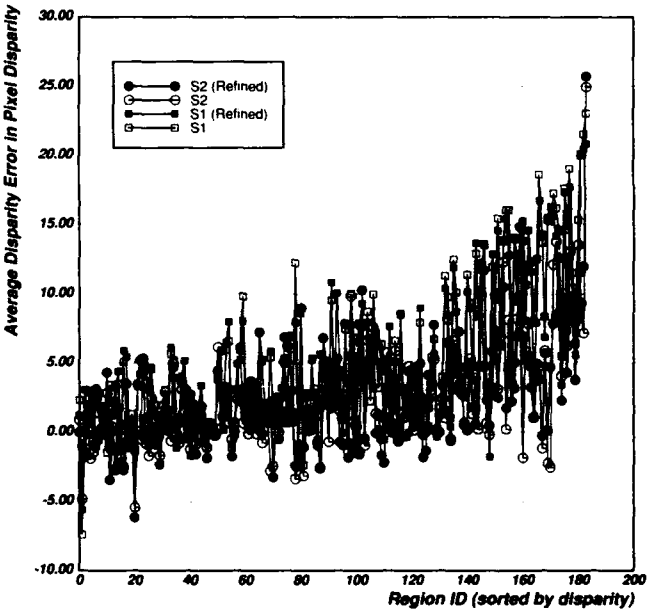


Figure 40: GAO1: Average Disparity Error by Building

ideal situation, no additional information would refine the disparity estimates; it would merely verify the truth of the disparity map.

Many improvements can be obtained by the use of better segmentations and scoring functions, and by addressing the assumption that only flat horizontal surfaces are responsible for the imaged radiometry and by using a more sophisticated surface model such as non-horizontal planar surfaces or quadratic surfaces. Finally, it seems feasible that multispectral data could be integrated by similar techniques. The information fusion approaches described here provide a means for data integration that may prove useful in other aspects of scene interpretation.

6. Acknowledgments

We would like to thank our colleagues in the Digital Mapping Laboratory for an interesting and congenial working environment. Jeff Shufelt provided helpful comments and suggestions on an earlier draft of this paper. Steve Cochran helped in the generation of the performance evaluation data.

References

[Barrow and Tenenbaum 78]
Barrow, H. G. and Tenenbaum, J. M.
Recovering Intrinsic Scene Characteristics from Images.
In Hanson, A. R. and Riseman, E. M. (editor), *Computer Vision Systems*, pages 3-26. Academic Press, New York, 1978.

[Cochran & Medioni 89]
S. D. Cochran and G. Medioni.
Accurate surface description from binocular stereo.
In *Proc. of the Workshop in Interpretation of 3D Scenes*. 1989.

[Eastman and Waxman 87]
R.D. Eastman and A.N. Waxman.
Using Disparity Functionals for Stereo Correspondence and Surface Reconstruction.
CVGIP 39:73-101, 1987.

[Hoff & Ahuja 89]
William Hoff and Narendra Ahuja.
Surfaces from Stereo: Integrating Feature Matching, Disparity Estimation, and Contour Detection.
IEEE Transactions on PAMI 11(2), 1989.

[Hsieh 90]
Hsieh, Y., Perlant, F., and McKeown, D. M.
Recovering 3D Information from Complex Aerial Imagery.
In *Proceedings: 10th International Conference on Pattern Recognition, Atlantic City, New Jersey*, pages 136-146. June, 1990.

[Hsieh, et. al. 92]
Hsieh, Y.C., McKeown, D.M., and Perlant, F.P.
Performance Evaluation of Scene Registration and Stereo Matching for Cartographic Feature Extraction.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-14(1):to appear, January, 1992.

[Irvin 89]
R. B. Irvin and D. M. McKeown.
Methods for exploiting the relationship between buildings and their shadows in aerial imagery.
IEEE Transactions on Systems, Man and Cybernetics 19(6):1564-1575, November, 1989.

[Lucas 84]
Lucas, B. D.
Generalized Image Matching by the Method of Differences.
PhD thesis, Computer Science Department, Carnegie-Mellon University, July, 1984.

[Luo 90]
Wei Luo and Henri Maitre.
Using Models to Improve Stereo Reconstruction.
In *Proceeding of 10th ICPR*, pages 60-64. 1990.

[McKeown 84]
McKeown, D.M., Denlinger, J.L.
Map-Guided Feature Extraction from Aerial Imagery.
In *Proceedings of Second IEEE Computer Society Workshop on Computer Vision: Representation and Control*, pages 205-213. May, 1984.

[McKeown 86]
McKeown, D.M., McVay, C.A., and Lucas, B. D.
Stereo Verification In Aerial Image Analysis.
Optical Engineering 25(3):333-346, March, 1986.

[Mohan & Nevatia 89]
R. Mohan and R. Nevatia.
Using Perceptual Organization to Extract 3-D Structure.
Tran. on PAMI 11(11), 1989.

[Ohlander 78]
Ohlander, R. B., Price, K., and Reddy, D. R.
Picture Segmentation Using a Recursive Region Splitting Method.
Computer Graphics and Image Processing 8:313-333, 1978.

[Poggio 88]
T. Poggio.
The MIT Vision Machine.
In *Proc. Image Understanding Workshop*. 1988.

[Shufelt and McKeown 90]
Shufelt, J., McKeown, D. M.
Fusion of Monocular Cues to Detect Man-Made Structures in Aerial Imagery.
In *Proceedings: IAPR Workshop on Multisource Data Integration in Remote Sensing*. June, 1990.
Also available as CMU Technical Report CMU-CS-90-194.

[Yakimovsky 76]
Yoram Yakimovsky.
Boundary and Object Detection in Real World Images.
Journal of the ACM 23(4), 1976.

Tools for Acquiring Spatial and Functional Knowledge in Aerial Image Analysis

Wilson A. Harvey, Matthew Diamond, and David M. McKeown, Jr.

Digital Mapping Laboratory
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

In order for rule-based interpretation systems to be at all practical it is important that there are tools available to create, maintain, and evaluate the system's knowledge base. This paper describes a suite of tools for the SPAM image interpretation system, that extends our previous research in this area [McKeown, et. al. 89]. These tools implement methods for interactive and automatic knowledge acquisition and knowledge-base evaluation. In addition, recent interpretation results from the SPAM system will be discussed.[1]

1. Introduction

SPAM is a production system architecture for the interpretation of aerial imagery with applications to automated cartography and digital mapping [McKeown, et. al. 85, CSReview 89]. It tests the hypothesis that the interpretation of aerial imagery requires substantial knowledge about the scene under consideration. Knowledge about the type of scene, whether airport, suburban housing development, or urban city, aids in low-level and intermediate level image analysis, and will drive high-level interpretation by constraining search for plausible consistent scene models. SPAM has been applied in two task areas: airport and suburban house scene analysis. In this section we describe the SPAM architecture, then briefly discuss related work in knowledge acquisition and knowledge-based vision.

1.1. Background: The SPAM Architecture

As with many computer vision systems, SPAM attempts to interpret the 2-dimensional image of a 3-dimensional scene. A typical input image is shown in Figure 1. The particular goal of the SPAM system is to interpret an image segmentation, composed of image regions, as a collection of real-world objects. For example, the output for the image in Figure 1 would be a model of the airport scene, describing where the runway, taxiways, terminal-building(s), etc., are individually located. SPAM uses four basic types of scene interpretation primitives: *regions*, *fragments*, *functional-areas*, and *models*. SPAM performs scene interpretation by transforming image *regions* into scene *fragment* interpretations. It then aggregates these fragments into consistent and compatible collections called *functional-areas*. Finally, it selects sets of functional-areas to form *models* of the scene.

As shown in Figure 2, each interpretation phase is executed in the order given. SPAM drives from a local, low-level set of interpretations to a more global, high-level, scene interpretation. There is a set of hard-wired productions for each phase that control the order of rule executions, the forking of processes, and other domain-independent tasks. However, this "bottom-up" organization does not preclude interactions between phases. For example, prediction of a fragment interpretation in *functional-area* (FA) phase will automatically cause SPAM to reenter *local-consistency check* (LCC) phase for that fragment. Other forms of top-down activity include stereo verification to disambiguate conflicting hypotheses in *model-generation* (MODEL) phase and to perform linear alignment in *region-to-fragment* (RTF) phase.

The first phase of SPAM is called region-to-fragment (RTF). This is a traditional heuristic classification process, using knowledge about the classes of features that occur in the scene to map a segmentation to a set of interpretations. Local properties of the segmentation, such as shape, texture, or height, are used to decide on which interpretations to generate. Examples of the type of knowledge used in region-to-fragment would be *runways are typically 50 to 80 meters wide*, or *houses are 8 to 10 meters high*.

The second phase of SPAM is called local-consistency

[1]This research was sponsored by the Air Force Office of Scientific Research under Contract AFOSR-89-0199. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Air Force Office of Scientific Research, or of the United States Government.

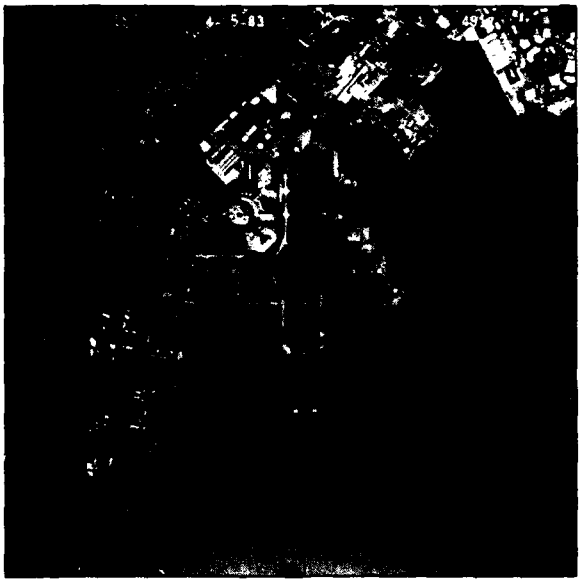


Figure 1: Aerial image of San Francisco Airport.

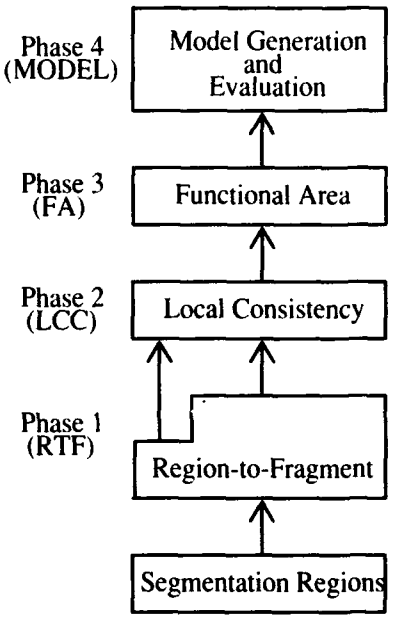


Figure 2: Interpretation phases in SPAM.

check (LCC). This phase performs a modified constraint satisfaction between the interpretations generated in region-to-fragment. The knowledge in this phase consists of the constraints between the different classes of objects. An example constraint would be *runways have perpendicular taxiways*. There are many such constraints in the system, with each constraint providing weak support to the participating hypotheses. Thus, it is not the action of a single constraint, but the collective action of several constraints which causes two interpretations to be found consistent with one another.

The functional-area (FA) phase groups together those interpretations that support one another, where support is computed from the results of the previous phase. A functional-area is defined as a group of interpretations that are similar in function and are in close physical proximity to one another. For instance, a terminal functional-area is defined to contain only terminal-building, parking-lot, road, and parking-apron interpretations. It is physically represented by the convex hull of the features in the functional-area.

Finally, the model-generation (MODEL) phase uses the functional-areas and combines them based on a number of heuristics, including number of conflicts, number of supported interpretations, and area of coverage. Conflicts are identified and resolved, either using support within the context of the model, or by invoking some process which would provide additional knowledge. For instance, if in the context of a model a region of the image was interpreted as both a taxiway and a hangar-building, a stereo process could be invoked to help resolve the conflict based on the region's height estimate. Multiple models are commonly generated and these models can be used as contexts for further processing.

1.2. Related Work

There has been a large body of work in traditional knowledge-based systems to support knowledge acquisition and validation [Gupta 91]. Typical issues include the validation of knowledge-based systems, checking for completeness and consistency in knowledge bases, and system performance evaluation. The application areas are broad, spanning medical diagnosis, design, process control, and configuration analysis. However, little work in this area has been focused within the context of computer vision tasks. One explanation is that there are few end-to-end knowledge-based systems for computer vision in the literature. A corollary is that those knowledge-based systems that have been built are "one-person" systems that do not survive much past the thesis defense. Some of the earliest "high-level" vision systems applied to aerial image analysis include Matsuyama's system [Matsuyama 80], the Sigma system [Hwang 84], and SPAM [McKeown, et. al. 85]. More recently, researchers have explored the use of generic knowledge [Huertas, et. al. 89] and investigated the formalization of knowledge to support high-level vision [Strat & Smith 88].

Our previous work in this area [McKeown, et. al. 89] focused on the use of compilation tools to translate a high-level schema based constraint representation into OPS5 productions. We also developed static analysis tools to aid in the display and debugging of the model descriptions generated by SPAM. As a result of the long term use of this system, our research focus has

shifted toward issues in automating portions of the knowledge acquisition process and in the development of tools to aid in the diagnostic analysis of finer levels of system behavior. In the following section we discuss recent work in these areas.

This paper describes recent work with the SPAM image interpretation system. Sections 2 and 3 discuss revisions and additions to our knowledge acquisition and analysis tools. Experimental results from each phase of processing in SPAM are presented and discussed in Section 4. Finally, Section 5 discusses current and future research.

2. Issues in Knowledge Acquisition for SPAM

In our previous work in knowledge compilation SPAM was separated into domain-dependent and independent parts to facilitate generalization, as well as to aid in our ability to maintain the system. The system is generated from a knowledge representation that is not dependent on the implementation language (which, in the current SPAM system, is OPS5). With the knowledge-base decoupled from OPS5, it is easier to create and maintain the large knowledge-bases needed to analyze complex scenes.

Tables 1 and 2 summarize the types of knowledge used in SPAM, the problems encountered in acquiring and using this knowledge, and the tools developed to address these problems. Several tools appear to be missing (e.g., FACHK). These may be added as necessary as the development of this line of research continues. Each of the currently implemented tools will be discussed in detail in the following sections.

2.1. Knowledge Acquisition of Spatial and Structural Constraints

RTFCHK, a knowledge acquisition tool for the first phase of SPAM, allows the user to interactively study and modify schemas encoding SPAM's shape knowledge. Correctness of individual rules is determined by comparing their results against the ground-truth interpretations of a scene. The results are sorted by correctness and by pass/fail into four displays. Figure 3 shows a confusion matrix generated from applying a runway rule to data from Washington National Airport. The matrix display shows true positives in the upper left, false negatives in the upper right, false positives in the lower left, and true negatives in the lower right. A rule containing constraints that are too restrictive will result in many false negatives, while loose constraints will cause many false positives. A "perfect rule" will have no false positives or negatives. Upon viewing these results the user may edit the schema and test it again.

RTFCHK was employed to improve a set of RTF schema that had been generated automatically from ground-truth files. The result was a set of RTF schema that exhibited improved performance over all of our airport test scenes (see discussion in Section 4.1). It is accepted that shape-classification alone cannot accurately classify complex scenes, but the RTF schemas produced using this tool provide better sets of interpretations for the later phases of SPAM.

A tool very similar to RTFCHK, called LCCCHK, aids the user in the knowledge acquisition process for the second phase of SPAM. For this phase, pairs of regions are tested against one another for attributes like closeness and perpendicularity. LCCCHK allows single constraints to be applied to classes of objects, and the results are compared against the ground-truth. Again, the user is presented with a confusion matrix display from which they may interactively inspect the results and/or make modifications to the knowledge base.

2.2. Automatic Shape Constraint Generation

One way for a user to develop new RTF schema is to allow them to measure many example regions. As more examples are viewed, the user can confidently assign reasonable values for the shape-attribute constraints. Given that we have generated several ground-truth datasets containing many examples of every class, this process can be automated.

RTFANALYZE is a tool that creates new RTF schema when given one or more scenes with ground-truth. It statistically analyzes the associated ground-truth segmentation to produce values, by class, for the various shape attributes. Currently, the process is exhaustive in that every attribute is computed for every class.

Given sufficient examples, RTFANALYZE produces schema that perform well across a range of airports[2]. The automatic schema generator produces effective rules much faster than a user can interactively, but often manual adjustments will improve a rule's performance. The combination of automatic generation followed by manual adjustment is currently our normal mode of operation for producing RTF schema.

2.3. Semi-Automatic Structural Constraint Generation via Functional Ground Truth

Much of SPAM's effort in hypothesis refinement is performed in the LCC phase. Because the space of possible constraints is large, manual methods of knowledge acquisition are inadequate. Automatic or semi-automatic methods for augmenting LCC knowledge are not only useful, but they become necessary to ensure that the complexity of the scene to be interpreted is captured.

The combinatorics of the second phase make

[2]Because the process is statistics based, this qualifier depends on how representative the examples are of the target class. Of course, other machine learning techniques, such as decision trees or clustering, can be used to help solve this problem.

Phase	Tool(s)	Kind(s) of Knowledge	Difficulties Encountered Using this Knowledge
RTF	RTFCHK, RTFANALYZE, FAANALYZE	Shape	Difficult to visualize using attribute/value representation; Large variations across scenes; Can require many examples; Shapes of object classes aren't unique.
LCC	LCCCHK, FAANALYZE	Local geometric	Context dependent; Many weak constraints required.
FA	FAANALYZE, SPAMEVALUATE	Grouping	Subjective functional definitions somewhat ambiguous; Very dependent on knowledge in LCC.
MODEL	SPAMEVALUATE	Global geometric, Conflict	Many knowledge sources (from different levels of processing) are required.

Table 1: Types of Knowledge Used in SPAM.

Tool	Phase(s)	Interaction	Problems Addressed
RTFCHK	RTF	Textual, Graphical	Allows graphical evaluation of shape constraints.
RTFANALYZE	RTF	Textual	Compiles shape constraints from examples.
LCCCHK	LCC	Textual, Graphical	Allows graphical evaluation of local geometric constraints.
FAANALYZE	RTF, LCC, FA	Graphical, Textual	Generates shape and local geometric constraints from examples using contexts
SPATS	All	Textual	Provides statistical feedback on SPAM's performance
SPAMEVALUATE	All	Graphical, Textual, Static graphical	Allows interactive graphical and textual access to all phases of SPAM results.

Table 2: Summary of Tools for Knowledge Acquisition in SPAM.

exhaustive observation of example scenes a very lengthy process, as every pair of regions must be tested using every constraint across several data sets. Furthermore, the LCC constraints would be extrapolated based on the observed relationships between arbitrary pairs of regions. This makes little sense when regions are unrelated; for example, a taxiway is usually perpendicular to an adjacent runway, but has no consistent relationship to distant runways.

In order to reduce the combinatorics and to be more precise about what context is used in the analysis, we decided to analyze only the structural relationships between elements of each functional-area type. This assumes that interactions between objects of different functional-areas is weak. The schema produced in this manner make more sense because they are derived from local relationships. Generating LCC schema automatically requires example (ground-truth) functional-areas.

FAANALYZE is a tool for semi-automatically acquiring and analyzing ground-truth functional-areas. It has an interactive mode that allows the user to specify a functional-area by clicking on its constituent regions. The system only allows the user to add regions which belong to the correct functional-area type. For

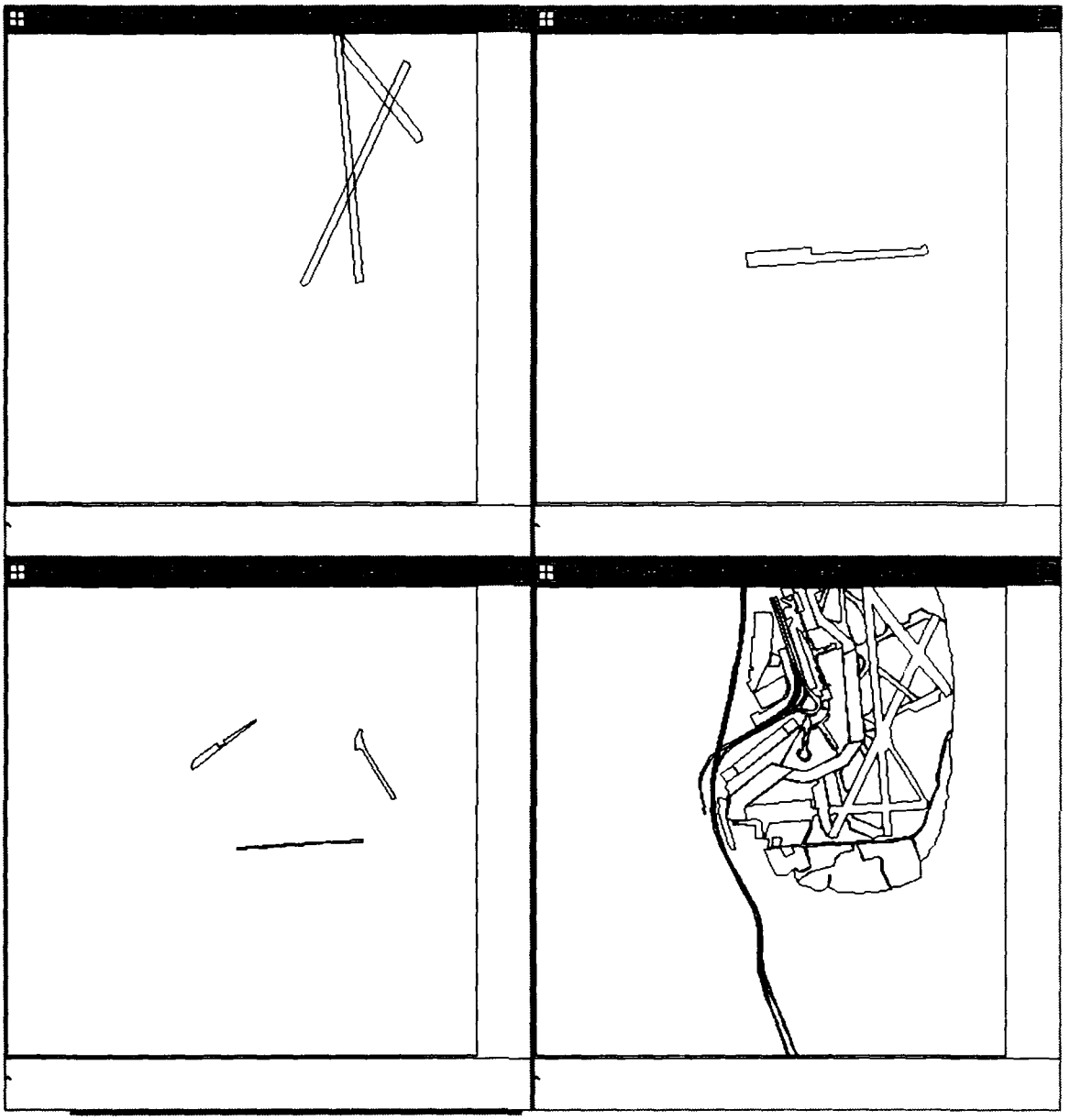


Figure 3: RTFCHK display after single schema has fired on ground-truth segmentation from Washington National Airport.

example, when entering a hangar functional-area, the user may choose only hangar buildings, roads, grassy-areas, and parking-aprons. Figure 4 is a snapshot of the display as a user adds regions to an example hangar functional-area.

Once ground-truth functional-areas have been created, FAANALYZE performs a statistical analysis, recording the observed values of the LCC constraints. Choosing which constraint types (distance, orientation) to use is a difficult problem, and is currently performed by the user. We are investigating the use of observed geometric distributions to decide which constraints perform best. As discovered with the RTF phase, a manual post-processing phase helps to improve the quality of the constraints generated automatically.

2.4. Acquiring Relative Shape Constraints

The program FAANALYZE addresses two issues. First, we wanted to generate LCC schema automatically by observing the relationships between region-types within functional-areas, as discussed previously. Second, we wanted to explore *relative shape constraints*.

We define relative shape attributes as the ratio between an attribute of functional-area seed-region and the same attribute for an element of the functional-area.

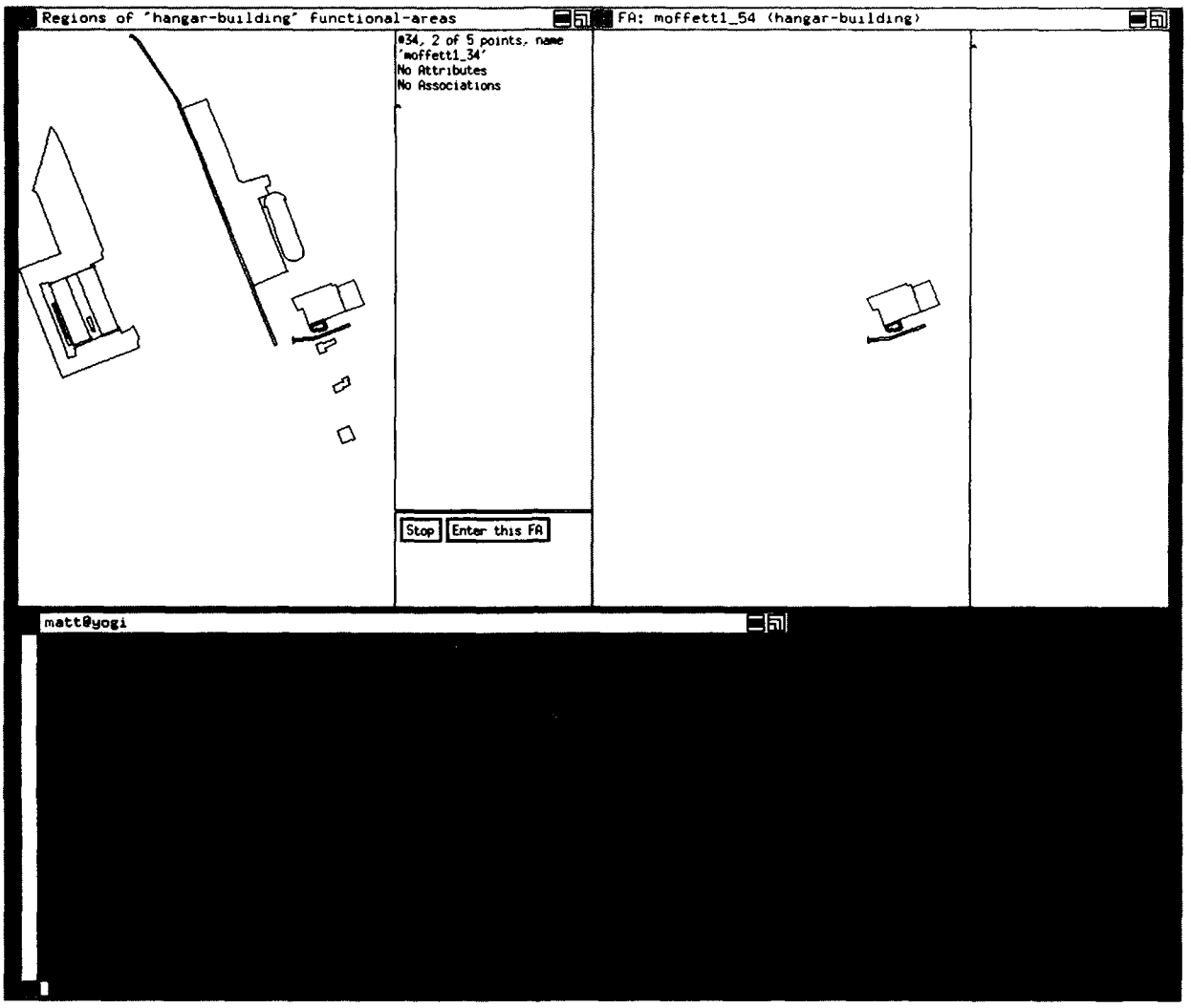


Figure 4: FAANALYZE: Selecting regions to build an example functional-area.

For instance, we hypothesized that if the seed region was larger than average, then one might expect the elements around it to be larger, as well. Several working-memory elements representing relative shape constraints for hangar-buildings and runways are shown in Figure 5. Unlike the first phase of SPAM, which tests the absolute shape-attributes of regions, the relative constraints take into account the relative size of the airport or subsets of it. The numbers, therefore, are unitless.

Generating relative shape constraints and LCC schema automatically requires example functional-areas. Once sample functional-areas have been created, FAANALYZE analyzes them, recording the observed values of both the relative shape constraints and the LCC geometrics. The output is in the form of working-memory elements and LCC schema. These are used to generate a new SPAM system.

The relative shape constraints can be used in the following three ways:

- To filter incorrect hypotheses from each functional-area;
- As a measure of confidence for entire functional-areas; if the ratios are outside the expected ranges, then either the seed region of the functional-area is incorrect or many constituents of the functional-area are wrong;
- To filter out unlikely hypotheses *before* the LCC phase fires, reducing the number of competing hypotheses in the LCC phase.

The relative shape constraints do, in fact, filter out hypotheses from functional-areas. However, we found that the local-consistency phase also filters many of the same hypotheses. We investigated using the relative shape-constraints to evaluate functional-areas as a whole, but the difference between a correct and incorrect functional-area was found to be difficult to quantify. This leaves only the third use, for filtering

```
(make fa-constraint  ^fa-context hangar-building
   ^from-class hangar-building  ^to-class road
   ^attribute  area       ^minimum 0  ^maximum 1818693  ^average 442)
(make fa-constraint  ^fa-context hangar-building
   ^from-class hangar-building  ^to-class tarmac
   ^attribute  perimeter  ^minimum 0  ^maximum 73371    ^average 728)

(make fa-constraint ^fa-context runway
  ^from-class runway           ^to-class taxiway
  ^attribute  area       ^minimum 1  ^maximum 157      ^average 9)
(make fa-constraint  ^fa-context runway
   ^from-class runway           ^to-class taxiway
   ^attribute perimeter   ^minimum 4  ^maximum 170      ^average 16)
```

Figure 5: Example relative shape constraints.

out hypotheses before the LCC phase fires. Because these constraints almost duplicate a subset of the LCC phase's results, we could use the relative shape-*constraints* without changing the final results. Since relative shape-constraints are faster than the LCC geometrics, this should reduce SPAM's total running time.

3. Analysis and Evaluation Tools

Post-run analysis and evaluation provides important feedback about the correctness of the knowledge-base and ways in which it can be improved. Some of our older tools, such as SPATS [McKeown, et. al. 89], produce statistical summaries of the performance of the SPAM system. Static run analysis can provide good summaries of system performance, allowing a user to easily gauge the aggregate affects of modifying a system's inputs. However, it can be difficult reasoning from this analysis about the specifics of why a result came to be. For example, knowing that a certain number of taxiways failed to pass a particular rule provides no information as to which taxiways failed, or why. While there is currently no automatic diagnostic tool for a SPAM user to consult, it seemed most natural to provide a graphical tool that would allow the user to interact with the results from a SPAM run. Our experience has found SPAMEVALUATE to be an effective diagnostic tool.

3.1. Interactive Performance Analysis

SPAMEVALUATE provides an interactive medium for browsing SPAM results and evaluating proposed modifications. It generates run statistics that augment those already generated by SPATS. It produces exhaustive and summary output. *Dumps* are a human-readable reproduction of the results of each phase; *summaries* are statistical synopses of the results. For example, the dump of the RTF phase lists each region, along with every RTF schema it passed. The summary, on the other hand, is a chart showing the breakdown of correct interpretations versus incorrect, allowing the user to infer which interpretations tend to be mistaken for one another. Statistics are available for the first 3 phases of SPAM; a sampling is shown in Figure 6.

Using the graphical mode of SPAMEVALUATE, the user can select regions by clicking with a mouse and then display very specific information culled from the SPAM run. The user can jump from phase to phase, observing how the results from each of the phases interact. The Figure 7 shows, in overlapping windows, the various phases of results as the user displays each in turn. Since the results are displayed graphically, the user gets a more intuitive feel for the meaning of the results. For example, a statistical summary might show that many taxiways are failing, but the graphical display would show that the failing taxiways tend to be longer than average, implying that the problem might lie in the RTF phase.

Another feature of SPAMEVALUATE is the ability to invoke schema of the first and second phases on individual regions, similar to the functionality provided by RTFCHK and LCCCHK. Since tracing through the results often suggests possible deficiencies in the current ruleset, the user can test such conclusions by invoking individual schema on any chosen region. The rule fires in a verbose mode, showing which constraints passed or failed, and by how much, as seen in Figure 8.

In practice, SPAMEVALUATE has been invaluable in our evaluation of the SPAM system. Sometimes a problem is solved by adjusting a constraint, or by adding a new schema to express a new piece of knowledge. SPAMEVALUATE also facilitates access to information necessary to study new methods of processing in the (more complex) functional-area and model phases. It is useful to be able to reason backwards from an interpretation result containing an error to the specific piece of knowledge causing that error. For instance, we recently changed the FA phase to remove competing hypotheses from within a functional-area, resulting in unique labels for the constituent objects (see Section 4.2.2). SPAMEVALUATE aided the algorithm development by permitting us to interactively trace back relationships from sets of

```
Generic Image: moffett1
Scenetype: airport
[RTF DUMP: interpretations generated by ground-truth region id]
    moffett1_0: navigational-aid parking-apron parking-lot grassy-area
                tarmac maintenance-building taxiway terminal-building
    moffett1_1: parking-lot taxiway road terminal-building
    moffett1_2: runway
        ...
[RTF STATS: class confusion matrix]
G.T. INTERPRETATION
|  SPAM INTERPRETATION -->
V      RW   TW   MRD  ARD  HWRD  CB   CHB  HB   MB   TB   CT   PA   PL   GA   TM   PMR  NVA  APJ
  RW    2    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  TW    0   36    0   15    0    0    0   10   15   27    0   16   28   15   17    0    8    0
 ARD    0    1    0    2    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  HB    0    5    0    1    0    1    0    9    7    7    0    6    7    7    7    0    6    0
  MB    0    1    0    0    0    0    0    0    1    1    0    1    1    1    1    0    1    0
  TB    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  PA    0    0    0    0    0    0    0    0    1    1    0    3    0    1    2    0    1    0
  PL    0    3    0    1    0    0    0    0    2    2    0    2    3    2    2    0    0    0
  GA    0    1    0    0    0    0    0    0    1    1    0    4    1   11    8    0    2    0
      ...
[LCC STATS: pairwise consistency table]
             HYPOTHESIS-PAIR (t-t)c (t-t)i (t-f)c (t-f)i (f-f)c (f-f)i Total#
      taxiway--runway          92%     ---     ---      1%     ---      0%       64
         road--road             2%     ---     ---     12%     ---     72%       50
hangar-buildi--road             4%     ---     ---     10%     ---     63%      108
hangar-buildi--hangar-buildi   21%     ---     ---      2%     ---     69%       92
terminal-buil--road             0%     ---     ---     10%     ---     85%      203
parking-apron--hangar-buildi    7%     ---     ---     13%     ---     67%      188
      ...
[FA DUMP: functional-area constituents by type]
FUNCAREA_77 (terminal)-- moffett1_1[TW]
         road: moffett1_32*
  parking-apr: moffett1_48[GA] moffett1_47[GA]
  parking-lot: moffett1_0[TW]
FUNCAREA_76 (road)-- moffett1_1[TW]
  grassy-area: moffett1_0[TW] moffett1_48* moffett1_47*
      ...
[FA STATS: comparison of generated interpretations to ground-truth by FA type]
runway
        runway-- 2 of 2 correct
        taxiway-- 21 of 21 correct
        grassy-area-- 11 of 11 correct
        tarmac-- 0 of 1 correct (1 GA)
hangar
        road-- 4 of 26 correct (18 TW) (4 PL)
        hangar-building-- 28 of 97 correct (68 TW) (1 TM)
        parking-apron-- 8 of 49 correct (25 TW) (8 HB) (7 PL) (1 GA)
        tarmac-- 0 of 21 correct (3 MB) (18 GA)
      ...
```

Figure 6: Selected statistics generated by SPAMEVALUATE.

functional-areas to LCC constraints between specific pairs of objects.

4. Experiments and Results

One tangible result of an image interpretation system should be the generation of a labeled set of segmentations. For SPAM, this labeled set includes the context in which the interpretations were decided upon (the model(s) and their constituent functional-areas), as well as the set of interpretations and related consistency information. However, a task as difficult and complex as aerial image interpretation allows many opportunities to contribute to the body of research knowledge, in addition to simply generating a final result. Evaluation of the effectiveness of different types of knowledge is an interesting as well as important element of our research goals. In this section, we present some experimental results from each of the phases of processing in SPAM. We discuss these results and evaluate the effects of the knowledge acquisition tools presented in the previous section.

4.1. Improving Shape and Spatial Constraints

As seen in Section 2, many of the tools developed for use with SPAM were created specifically for the purpose of improving the results emerging from a single processing phase. For example, the program RTFANALYZE allowed us to automatically generate sets of RTF schemas from statistics produced from our airport ground-truth database. Because the RTF phase

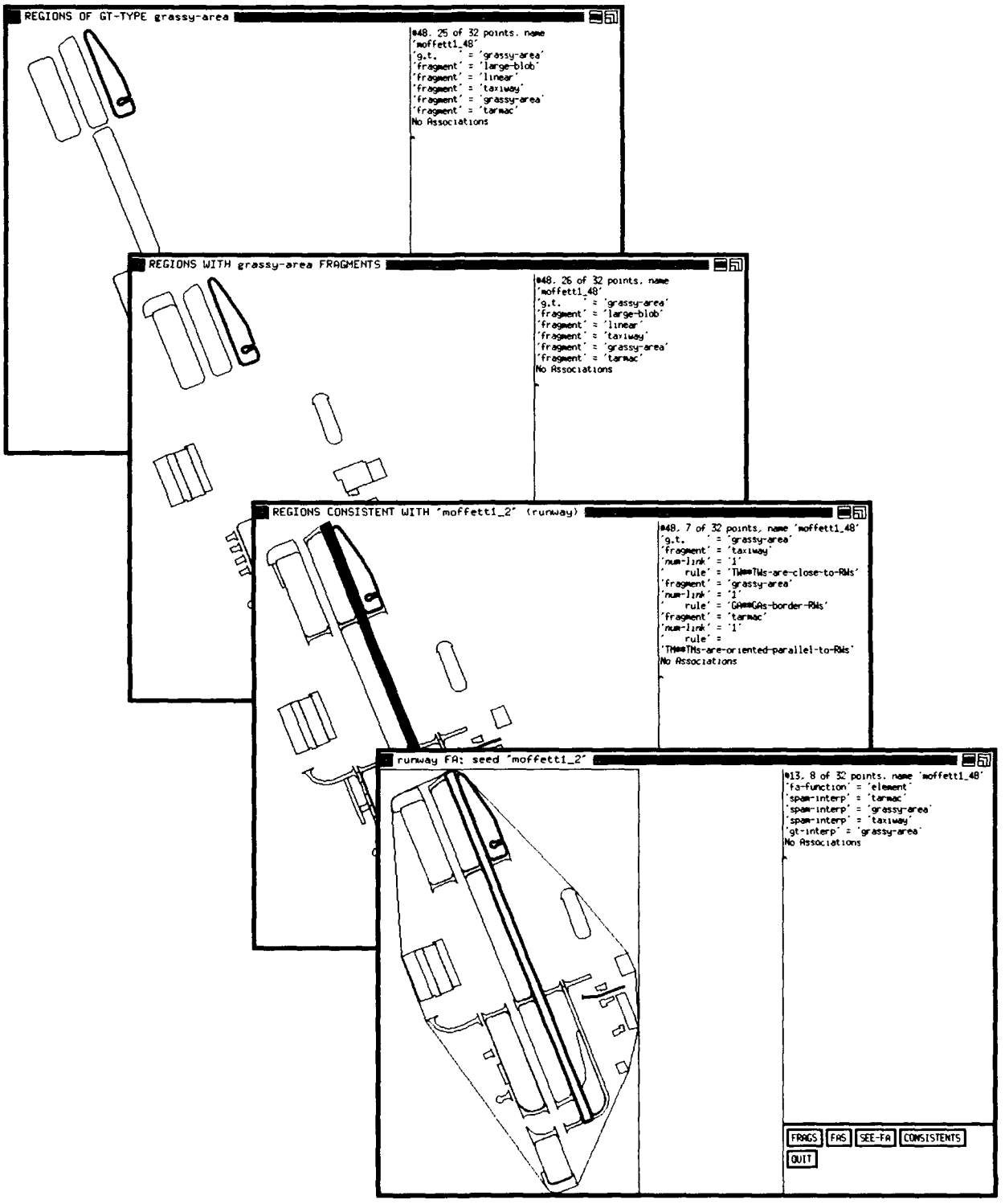
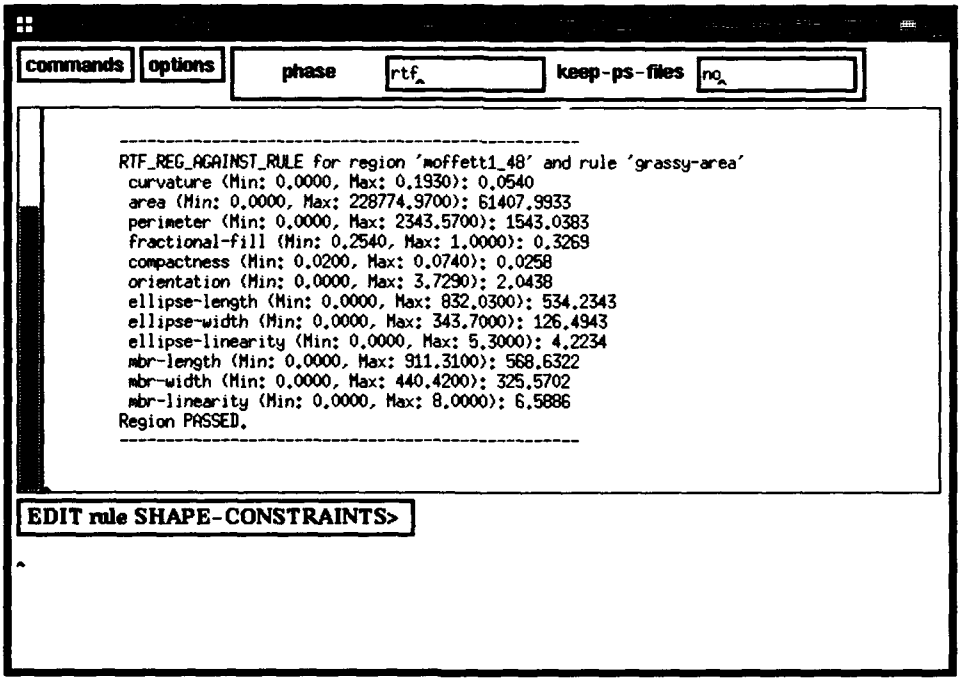


Figure 7: Various evaluation methods in SPAMEVALUATE.



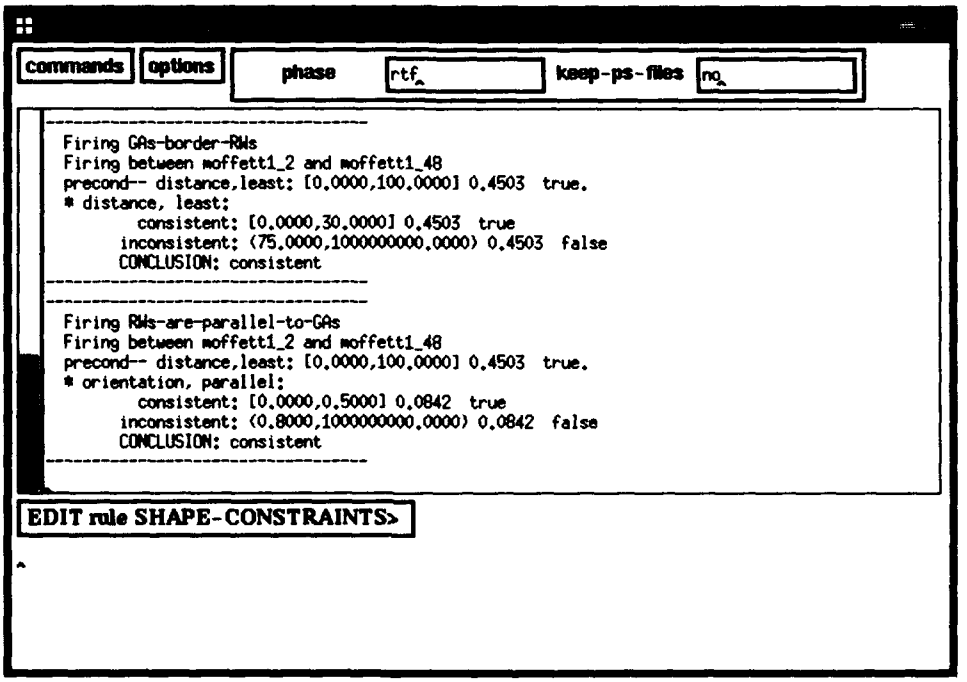


Figure 8: SPAMEVALUATE: Invoking RTF and LCC schema.

deals only with local properties of each segmentation, a brute-force approach is tractable. These automatically generated schema were found to improve the classification of certain object classes (e.g., taxiways) when compared with the hand-generated schema. Figure 3 shows these differences. For example, the number of true positives for taxiways increased from 11 to 28. However, the number of false positives also increased from 11 to 46. This trend is true, in general, across all object classes. The issue is that most of the false positives will not be found to be consistent with other hypotheses during SPAM's LCC phase. Therefore, it is more important for a larger number of true positives to proceed past RTF.

From evaluating the results of running SPAM on several different airport scenes, it became clear that knowledge in LCC was incomplete. This was most evident when examining functional-area results, as it was easy to see that certain object classes were incorrectly supporting (or not supporting) one another. For the LCC phase, quantitative result evaluation is more complicated because constraints are not universally true. For example, though distance constraints exist between hangar-buildings and roads (as classes), there may be no such relationship between a specific hangar-building/road pair. A ground-truth database of constraints to permit such an evaluation would be tedious time-consuming to generate (though not impossible). In lieu of generating such a database, we designed several experiments to assist us in evaluating the effectiveness of the LCC rules. Two of these experiments were:

- Replacing the output from RTF with a set of perfect interpretations;
- Supplementing the output from RTF with a set of perfect interpretations.

The set of perfect interpretations can be easily generated from the ground-truth. We can qualitatively evaluate the results of these experiments using our knowledge acquisition tools while also refining the existing knowledge base. We can then quantitatively evaluate the results of rerunning the SPAM system.

The first experiment helped to detect problems with lack of support between correct hypotheses (false negatives). Missing support would indicate either a lack of constraints or incorrect (too narrow) bounds on the existing constraints. SPAMEVALUATE and LCCCHK were used to evaluate proposed constraints as well as to locate and fix problems with existing constraint bounds.

The second experiment aided in identifying incorrect hypothesis support (false positives). As stated previously, SPAM assumes that *consistency* between two interpretations is determined by the action of multiple constraints. Therefore, it is expected that there will be many constraints falsely satisfied, and that they will be distributed randomly among the available hypothesis classes. However, if a rule has a very permissive bound, then many false positives occur, and the constraint poorly discriminates one class from another. Again, both SPAMEVALUATE and LCCCHK were used to identify useless constraints and to tighten bounds on permissive constraints.

In addition to manual modifications to the knowledge base, we used FAANALYZE to semi-automatically generate constraints for the LCC phase. FAANALYZE generates all possible geometric constraints and performs only simple pruning of the results. A significant portion of these constraints provide very little discrimination between consistent and inconsistent hypotheses, though several missing constraints were found. Therefore, creating LCC schema is still a task for a knowledgeable user, but FAANALYZE is still useful as an advisory tool, supplying the range of expected values once a geometric constraint has been selected.

4.2. Functional Groupings

Viewing the results of FA from a number of airports using SPAMEVALUATE revealed several problems, the most apparent of which were:

- Large functional-areas — The generated functional-areas covered large portions of the scene and included many regions.
- Multiple interpretations — Within a functional-area, a given region could have multiple conflicting interpretations.

Discussion of and solutions to these problems are reviewed below.

4.2.1. Focusing Functional-Areas

The ideal functional-area should include only the seed (or, generating) region and those regions consistent with the seed that surrounding it. SPAM requires small functional-areas for several reasons. One purpose of the functional-area is to provide an object whose set of constituents are relevant to one another, i.e., they exert some constraint on most of the other members of that set. If a functional-area is too large then bounds on constraints between members of that functional-area will tend to be less precise. Additionally, a large number of functional-area elements makes the process of deciding ambiguous interpretations computationally more difficult (see Section 4.2.2).

Using knowledge that was collected before the creation of our knowledge acquisition tools, the functional-areas generated by SPAM were too large, both in size and in number of elements. Some of this was corrected as a result of using these tools to improve both RTF and LCC results. However, enough problems remained that we felt it necessary to develop a method for focusing the application of constraints to local regions around our current area of interest.

A natural solution to this problem was to allow preconditions in local-consistency rules in order to

Class	# Ground Truth	Auto True Positives	Auto False Negatives	Auto False Positives	Auto True Negatives	Hand True Positives	Hand False Negatives	Hand False Positives	Hand True Negatives
runway	4	3	1	0	163	0	4	0	163
taxiway	39	28	11	46	82	11	28	11	117
road	2	1	1	20	145	0	2	3	162
terminal-building	9	4	5	54	104	2	7	23	135
hangar-building	17	11	6	56	94	5	12	12	138
parking-apron	0	0	0	55	112	0	0	59	108
parking-lot	0	0	0	74	93	0	0	98	69
grassy-area	45	43	2	57	65	35	10	14	108
tarmac	6	3	3	82	79	1	5	36	125

Table 3: Comparison of RTF confusion matrices for hand versus auto generation for San Francisco.

force each rule to focus on smaller portions of the scene. For example, we use a maximum distance test to prune away regions that are being considered for orientation tests. Because the functional-area is computed from the local-consistency information, this addition reduces the number of interpretations considered and, therefore, the size of the functional-area.

Example functional-areas, before and after addition of preconditions, are shown in Figures 9 and 10, respectively. The seed for these two functional-areas is the same (the runway region highlighted in bold), and the interpretations that are incorrect or ambiguous are shown in grey. The change in size is obvious, and one can see that not only is the new functional-area more focused, but it contains no incorrect or ambiguous interpretations when compared to the ground-truth. As expected, the preconditions are providing a context around each interpretation within which we can apply the spatial constraints.

4.2.2. Ambiguous Interpretations

Another problem we addressed was the resolution of ambiguous interpretations within a functional-area. Previously, we would delay resolution of such conflicts until MODEL phase, making the model-generation process more complicated. Making these decisions in FA seems more intuitive, as functional-areas emerging from this phase are, as a result, internally consistent.

The context provided by the functional-area is used to decide between two or more competing interpretations. Again, the solution that seemed most natural was to use more complicated LCC constraints. An example of such a constraint is *tarmacs are close to and located at the ends of taxiways*. These types of constraints could be implemented as functions of the binary constraints already used in LCC. This implies that a function involving a count of the number of positive and negative links could be an effective method of determining which of several hypotheses is most supported. This function should have the property that, when evaluated over all the supporting hypotheses, fewer multi-constraint successes should receive a higher confidence than many single-constraint successes.

We chose a function that totaled all positive links from unambiguous hypotheses (regions with a single interpretation in the context of the functional-area) for each ambiguous interpretation (regions with multiple interpretations within the context of the functional-area). Those unambiguous hypotheses supporting the conflict with more than a single satisfied constraint receive extra weight in the confidence computation. The interpretation with the most weight remained in the functional-area; any competing interpretations were filtered out.

Table 4 shows statistics compiled across all the filtered functional-areas generated from a single run. The chart shows the four airport functional-area types, broken down into their respective elements. Each element-type is followed by four columns showing, in order, the number of correct hypotheses of that type found in functional-areas, the number incorrect, and then the results of the filtering (number of correct/incorrect hypotheses filtered). The numbers indicate that, in most cases, the incorrect hypotheses are being filtered out more often than correct ones. For instance, every incorrect hypothesis of the runway functional-areas is filtered, and only two correct hypotheses are removed. This means that the filtering algorithm and the consistency information it uses are generally working well. A specific observation should be made with regard to terminal-building functional-areas. From the table it can be seen that parking-aprons within terminal-building functional-areas seem to be filtered more or less randomly. This is due in part to there

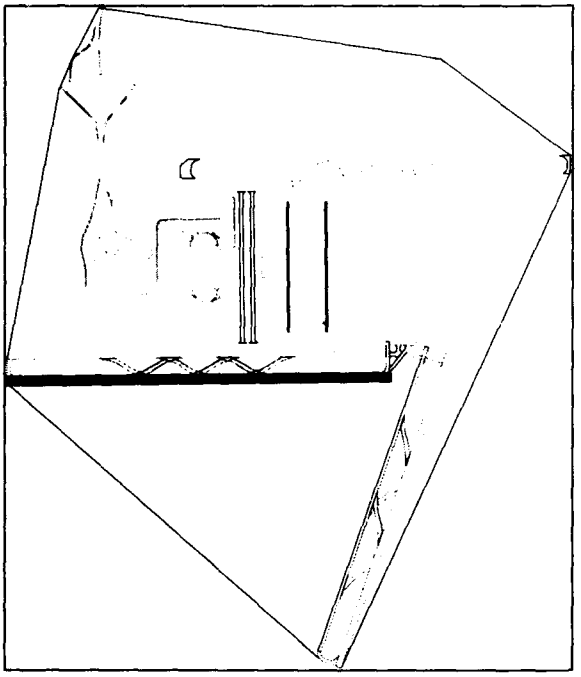


Figure 9: Functional-Area from Dulles International.

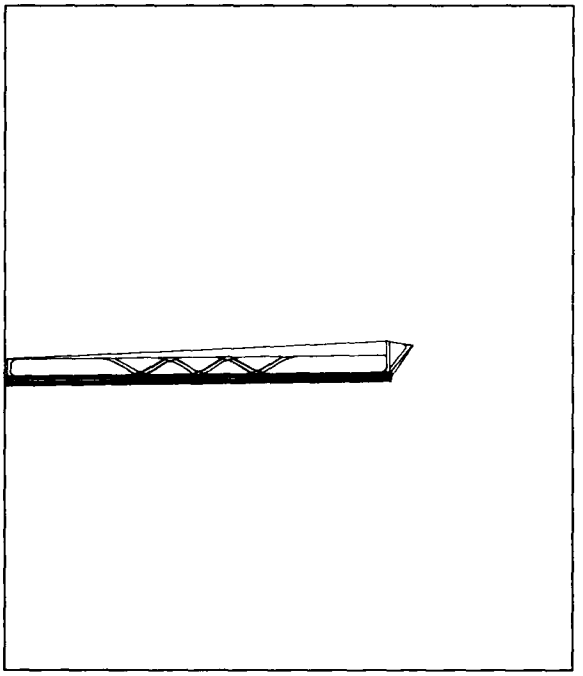


Figure 10: New Functional Area after Preconditions.

FA Type	Element Type	Correct Hypotheses	Incorrect Hypotheses	# Correct Hypotheses Filtered	# Incorrect Hypotheses Filtered
runway	runway	3	0	0	0
	taxiway	24	1	0	1
	grassy-area	20	6	2	6
	tarmac	0	19	0	19
hangar	hangar-building	75	319	0	0
	road	79	65	3	21
	parking-apron	11	219	1	117
	tarmac	19	236	0	140
terminal	terminal	2	105	0	2
	road	102	104	10	29
	parking-apron	16	335	14	281
	parking-lot	43	541	0	61
road	road	128	47	0	0
	grassy-area	94	126	0	29

Table 4: Dulles functional-area summary.

being such a small ratio of correct to incorrect terminal-building functional-area seeds. This is the fault of the consistency information for terminal-building functional-areas, not of the filtering algorithm.

4.3. Results for Model Generation

The final phase of SPAM must produce a consistent scene model by merging functional-areas while resolving any ensuing conflicts. Multiple models should be generated when competing models of the same scene are radically different in layout. Model generation is a hard problem, conceptually and combinatorially; the strictly algorithmic approach, generating all possible scene models, can be shown to be NP-complete (see Section 4.3.2). This section outlines three different methods that we have used for generating final scene models from functional-areas. These are:

1. Generating a tiling of the scene;
2. Generation of maximal cliques from a graph;
3. Using heuristic search.

We are experimenting with each of these methods, though the heuristic search method currently shows the most promise.

4.3.1. Tiling

One of our approaches to producing a scene model involves generating a tiling of the scene in question. Such an algorithm tries to choose those functional-areas that maximize the amount of the scene that is covered. This would be a trivial problem if it weren't for the fact that functional-areas overlap, and interpretations for those regions in the areas of overlap can conflict. These conflicts must either be avoided, or resolved intelligently, as they determine the overall goodness of the generated model. Therefore, constraints are applied which allow the system to intelligently choose which functional-areas can coexist.

The algorithm works as follows. Initially, the model is empty. A model *seed* (an initial functional-area) is chosen, either automatically or by the user, and consecutive functional-areas are added to this seed based on maximizing the number of new regions, minimizing the number of hypothesis conflicts, and maximizing support from existing functional-areas. The process terminates when a given percentage of the input regions have been used in the model.

This procedure efficiently generates a single model which is very dependent on the quality of the initial seed functional-area. We believe this dependence could be lessened by creating better functional-area constraints. Further development would emphasize embedding this algorithm in a framework that would allow the ability to generate and rank multiple models.

4.3.2. Maximal Cliques

Another approach to model generation views each functional-area as a node in a graph, with arcs existing based on the compatibility of two functional-areas. Compatibility can be computed by applying constraints, such as those used in Section 4.3.1, to functional-area pairs. Once the arcs have been computed, models can be extracted from the graph by searching for maximal cliques, or by looking for cliques of a particular size. Algorithms for these problems exist, but their worst-case performance is known to be exponential in the number of inputs. In this case, the number of models generated can be exponential in the number of functional-areas. For our experiments, only the smallest sets of functional-area results could be used to generate models (no more than 40 or 50 functional-areas). Modifications to this method, such as adding domain knowledge to constrain the search for cliques, or reducing the numbers of nodes in the graph by merging functional-areas of the same type, could help make the process tractable.

4.3.3. Heuristic Search

Our most recent research in model generation uses heuristics to constrain which sets of functional-areas can coexist in the same model. A best-first search is performed in the space of possible functional-area groups. Conflicts between competing hypotheses from different functional-areas are enumerated, but are currently not resolved.

The key to directing the search for models efficiently and intelligently is the set of heuristics used. No attempt is made to rank or order the heuristics. Instead, all heuristics contribute equally to the functional-area's score, some in a positive manner and others negatively. The score improves if the functional-area adds new regions and/or support to the current model, is very compact, or covers a lot of area. The score is reduced if conflicts are added to the model. The score is normalized according to the number of regions in the functional-area in an attempt to allow functional-areas of varying sizes to participate equally in model-generation. Other heuristics include the density of the entire model, and the density of the conflicts. A tight group of conflicts is believed to be better than one with conflicts spread out all over the scene.

All the model heuristics discussed thus far are domain-independent. There are instances where one would like to bring domain-dependent constraints to bear, such as constraints between functional-areas, much like the LCC phase evaluates consistency between pairs of regions. Examples of such constraints are *terminal-buildings are centrally located with respect to the runways* (in fact, there are functional reasons for the control-tower or terminal-building area to be more or less equidistant from each runway) and *hangar-buildings are not centrally located.* Our experiments with these constraints indicate that they are not good predictors and so cannot be used to intelligently generate and/or fix models, though they can be used to evaluate existing models. Therefore, our current scheme generates multiple models (without using scene dependent knowledge), then allows domain-dependent constraints to be used to rank the models produced. Further exploration of these types of constraints is material for future work.

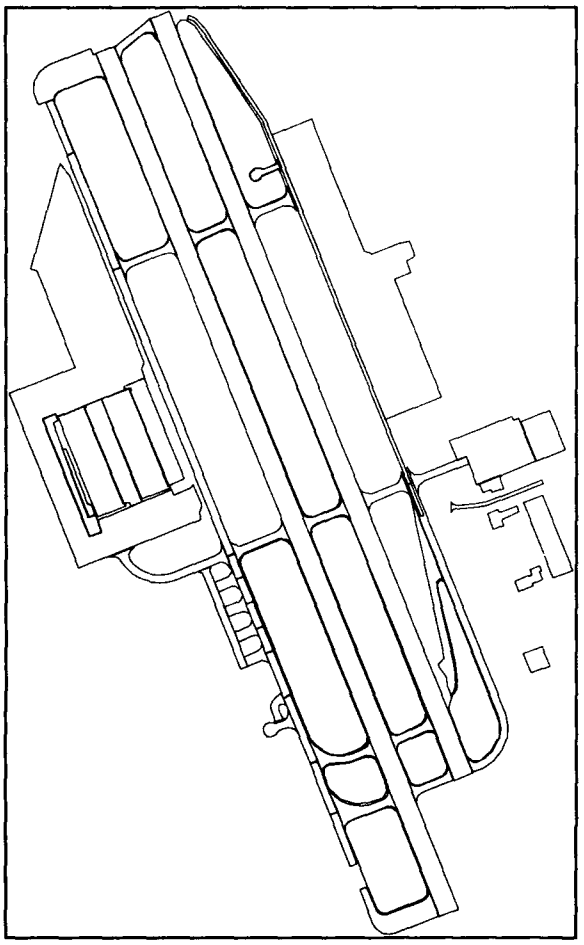


Figure 11: Model of Moffett AFB.

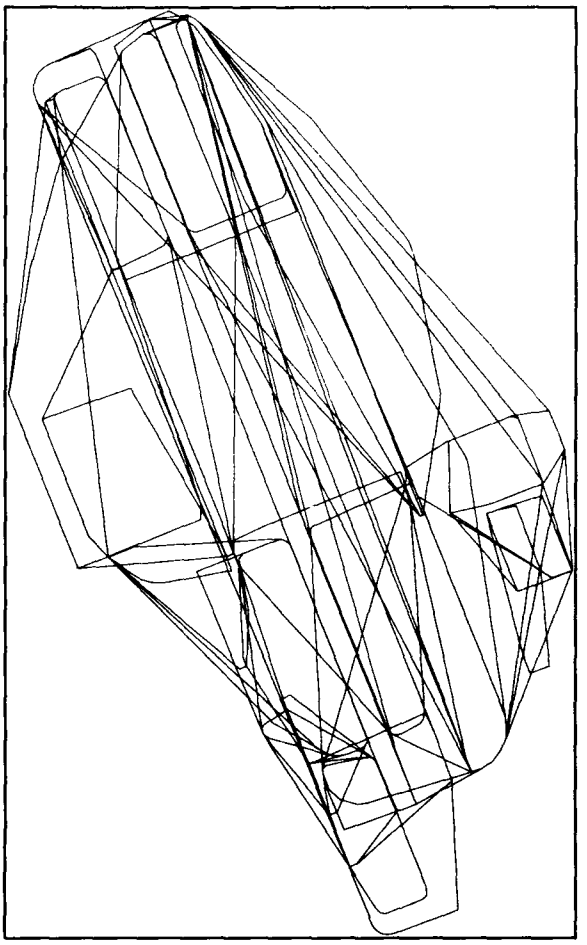


Figure 12: Functional-Areas in the Model.

5. Current and Future Work

SPAM supports a variety of research activity within the context of image understanding. It is a research vehicle for our knowledge acquisition work, experience with systems integration, and further research in task-level parallelism [Harvey, et. al. 90, Harvey, et. al. 91]. As such, there are many promising areas of future work. Several of these are outlined below.

As with any system, testing on many cases aids in finding implicit assumptions. More airport examples would not only help in refining SPAM's knowledge base, but this would allow exploration of categories within the airport domain. For example, it may not be as effective to have a "general" airport knowledge base as it would to have a separate knowledge base for each one of *military*, *international*, and *regional* airports.

In addition to other airports, different segmentation methods will likely provide SPAM with new challenges. We have, at various times, experimented with using segmentations obtained from automatic methods, such as some of the feature extraction systems being developed in our group [McKeown 88, Irvin 89, Shufelt & McKeown 90, Hsieh, et. al. 92]. Issues include how much of the knowledge has to be tailored according to the source of the segmentation, as well as what effects are produced by errors in the segmentation. Such feature extraction systems could also be used in the later phases of SPAM as domain experts, assisting in the disambiguation of conflicting hypotheses.

Expanding some of our work in automatic constraint generation for SPAM, such as was done with RTFANALYZE and FAANALYZE, is another promising area of future research. Applying clustering techniques may allow the system to automatically filter out constraints that result in little or no discrimination among competing hypotheses. One could augment such a system with interactive capabilities so that it could suggest rules to a user and allow that user to reject or refine them.

In addition to improving the quality of the functional-areas produced by SPAM, we need to intelligently evaluate the coherence of each functional-area, as well

as to compare them to one another. One method for doing this is to examine the support for each hypothesis and compare it to a "randomly" supported hypothesis. If this or other methods are found to be effective, functional-areas can be ranked, unpromising functional-areas can be excised, and model-generation can proceed based on this ranking.

Finally, our work with the new methods for model-generation is still somewhat preliminary. In addition to more experimentation, developing a schema-based abstraction of the types of knowledge required for effective model generation would aid in the development of knowledge acquisition tools for this process.

6. Conclusions

Our ongoing research in the automated analysis of complex aerial imagery relies heavily on our ability to provide spatial and structural constraints, encoded as rules, to SPAM, our knowledge based interpretation system.

Large-scale knowledge-based vision systems require specialized tools for both knowledge acquisition and result evaluation. As we have seen, this research has focused on the interactive acquisition of spatial and functional knowledge as well as on fully automated techniques. We have shown preliminary results based on our experiments with airport scenes and also demonstrated the importance of manually compiled ground truth scene segmentations to support rigorous performance analysis tools. The ability to generate accurate evaluations of system performance guides our research along paths that are likely to prove most productive.

Further research is needed in basic computer vision and image understanding, the architecture of knowledge-based systems, and their integration with spatial databases. We believe that our research addresses some of the most important issues in these areas and that we are progressing toward the development of competent automated scene analysis systems.

Acknowledgements

We thank the members of the Digital Mapping Laboratory for many helpful discussions. In addition, Jeff Shufelt and Milind Tambe gave us many valuable comments on an earlier draft of this paper.

References

[CSReview 89]
David McKeown, Aviad Zlotnick, Frederic Perlant, Yuan Hsieh, Wilson Harvey, and Matthew Diamond.
Research in Digital Mapping.
Computer Science Research Review 88/89:39-63, 1988/1989.

[Gupta 91]
Gupta, U.
Validating and Verifying Knowledge-Based Systems.
IEEE Computer Society Press, 1991.

[Harvey, et. al. 90]
Harvey, W., Kalp, D., Tambe, M., McKeown, D. and Newell, A.
The Effectiveness of Task-Level Parallelism for High-Level Vision.
In *Proceedings of the 2nd ACM/SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, pages 156-167. March, 1990.

[Harvey, et. al. 91]
Harvey, W., Kalp, D., Tambe, M., McKeown, D. and Newell, A.
The Effectiveness of Task-Level Parallelism for Production Systems.
Journal of Parallel and Distributed Computing (to appear), December, 1991.

[Hsieh, et. al. 92]
Hsieh, Y.C., McKeown, D.M., and Perlant, F.P.
Performance Evaluation of Scene Registration and Stereo Matching for Cartographic Feature Extraction.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-14(1):to appear, January, 1992.

[Huertas, et. al. 89]
Huertas, A., Cole, W., & Nevatia, R.
Using Generic Knowledge in Analysis of Aerial Scenes: A Case Study.
In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1642-1648. Morgan Kaufmann Publishers, Inc., August, 1989.

[Hwang 84]
Shang-Shouq Vincent Hwang.
Evidence Accumulation for Spatial Reasoning in Aerial Image Understanding.
PhD thesis, University of Maryland, 1984.

[Irvin 89]
R. B. Irvin and D. M. McKeown.
Methods for exploiting the relationship between buildings and their shadows in aerial imagery.
IEEE Transactions on Systems, Man and Cybernetics 19(6):1564-1575, November, 1989.

[Matsuyama 80]
Matsuyama, T.,
A Structural Analysis of Complex Aerial Photographs.
Technical Report, Department of Electrical Engineering, Kyoto University, Kyoto, Japan, April, 1980.
Ph.D Thesis.

[McKeown 88]
McKeown, D.M. and Denlinger, J. L.
Cooperative Methods for Road Tracking in Aerial Imagery.
In *Proceedings IEEE Computer Vision and Pattern Recognition Conference*, pages 662-672. June, 1988.

[McKeown, et. al. 85]
McKeown, D.M., Harvey, W.A. and McDermott, J.
Rule Based Interpretation of Aerial Imagery.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-7(5):570-585, September, 1985.

[McKeown, et. al. 89]
McKeown, D.M., Harvey, W.A., Wixson, L.
Automating Knowledge Acquisition For Aerial Image Interpretation.
Computer Vision, Graphics and Image Processing 46(1):37-81, April, 1989.

[Shufelt & McKeown 90]
Shufelt, J.A., and McKeown, D.M.
Fusion of Monocular Cues to Detect Man-Made Structures in Aerial Imagery.
Technical Report CMU-CS-90-194, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA., September, 1990.

[Strat & Smith 88]
Strat, T., & Smith, G.
Core Knowledge Systems: Storage and Retrieval of Inconsistent Information.
In *Proceedings of the DARPA Image Understanding Workshop*, pages 660-665. Morgan Kaufmann Publishers, Inc., April, 1988.

On the Uniqueness of Correspondence under Orthographic and Perspective Projections

Ronen Basri

M.I.T., A.I. Lab., NE43-792
545 Technology Square
Cambridge, MA 02139
E-mail: ronen@ai.mit.edu

Abstract

The task of shape recovery from a motion sequence requires the establishment of correspondence between image points. The two processes, the matching process and the shape recovery one, are traditionally viewed as independent. Yet, information obtained during the process of shape recovery can be used to guide the matching process. In this paper we review the constraints imposed on the correspondence by rigid transformations and extend them to objects that undergo general affine (non rigid) transformation (including stretch and shear), as well as to rigid objects with smooth surfaces. In all these cases corresponding points lie along epipolar lines, and these lines can be recovered from a small set of corresponding points. We discuss the potential use of epipolar lines in the matching process, and present an algorithm that recovers the correspondence from three contour images. The algorithm was implemented and used to construct object models for recognition.

1 Introduction

Correspondence is a process of relating information in one image to its equivalent in others. Using correspondence a vision system can infer the 3-D structure of the observed scene, an inference that is significantly more difficult to make from a single 2-D image. The establishment of correspondence is itself a difficult task. Various methods were developed in recent years to achieve correspondence for stereo vision [Baker & Binford 1981, Marr & Poggio 1979, Grimson 1980], motion analysis [Koenderink & Van Doorn 1975, Ullman 1979, Longuet-Higgins 1981, Hildreth 1984], and object recognition [Fischler & Bolles 1981, Grimson & Lozano Pérez 1984, Lowe 1987, Huttenlocher & Ullman 1987].

In motion analysis the stage of establishing correspondence is usually viewed as independent of the stage of shape recovery [Ullman 1978]. According to this view, the correspondence is determined so as to minimize the observed 2-D motion along the image sequence. No assumptions are made at this stage with respect to the shape of the moving objects or to the transformations they undergo. In this way correspondence can be found even for non rigid objects and when the images contain a number of objects moving differently.

The distinction between the two processes of correspondence and shape recovery is useful when the motion between the frames is relatively small, in which case a minimization process can resolve the correspondence correctly. When, however, "long range motion" is considered, minimization techniques often fail to find the correct correspondence. Information about the transformation may be used in these cases to guide the process of establishing correspondence.

An important application that requires correspondence under "long range motion" conditions is the construction of 3-D representations for object recognition. In this process shape information is accumulated over time until a complete model is constructed for the object. During this period the object may be observed in positions that significantly differ from one another. Yet, it is desired from this process to tolerate such differences.

The process of model construction is not only useful for building object-centered models, but also for constructing viewer-centered ones. A recognition scheme that uses viewer-centered representations was recently developed [Ullman & Basri 1991]. In this method an object is represented by a small number of its 2-D images together with the correspondence between the images. The appearance of an object from different viewpoints is predicted by the linear combinations of its model images. These predictions are exact for rigid objects. In both object-centered as well as viewer-centered cases the representations obtained are often more stable as the images used to construct the models are taken from viewing angles that are relatively distant from one another.

One assumption that is generally used in different vision applications such as motion and object recognition is that the objects observed are rigid. Huang & Lee [1990] have recently addressed the question of how rigidity affects the solution to the correspondence problem. They showed that under an orthographic projection the correspondence to points cannot be determined but up to straight lines (known as "the *epipolar lines* of the points"), and that four corresponding points determine the position of these lines. They did not specify any method to resolve the correspondence within the lines.

The epipolar line idea is not new. It is extensively used

in stereopsis, but rarely used in establishing correspondence in motion analysis. Bolles & Baker [1985] used epipolar lines to analyze motion sequences obtained by a translation along a straight line. Yachida [1986] and Ayache & Lustman [1987] used it in developing their trinocular stereovision algorithm. In this paper we examine the use of the epipolar lines in establishing correspondence for depth reconstruction. In the first part of this paper (Section 2) we review the theory behind the epipolar lines and the way to compute them from a small number of corresponding points. The formulation we use is somewhat different from that presented by Huang & Lee [1990], and we analyze the similarity and the differences between orthographic and perspective projection models. We show that epipolar lines exist even in more complicated situations, such as, when an object undergoes a general linear transformation (including stretch and shear), and when objects with smooth surfaces are considered. In the second part of this paper (Section 3) we show that the correspondence is not determined uniquely even when three or more images are given. Additional images can, however, be used to heuristically resolve the correspondence [Yachida 1986]. Together with additional constraints such a heuristic can, in practice, solve the correspondence problem. We have implemented this algorithm and the results were used in constructing object models for recognition.

2 Correspondence from Two Images

The correspondence problem discussed below is defined as follows. Given a pair of 2-D images, for every point in space that is projected to both images find its location in the two images. Often only feature points (such as, contour points) are considered. We examine this problem assuming the images differ by a rigid transformation. We consider two projection models, orthographic projection (with a uniform scale factor to compensate for depth changes) and perspective projection. We begin our discussion by introducing general properties for both projection models, and later prove these properties for each of the models separately. Finally, we extend these properties to more complicated cases, such as, objects that undergo general affine transformations (rather than rigid ones) and objects with smooth surface boundaries.

Our analysis consists of three steps:

1. We show that rigidity divides the images into sets of epipolar lines, their correspondence is determined by the transformation that separates the images, but the correspondences of points along the lines cannot be determined.
2. The epipolar lines can be recovered from a small set of corresponding points, four in the orthographic case and seven in the perspective case.
3. These results apply also to objects that undergo general affine transformation and to objects with smooth surfaces.

Proposition 1 establishes that in a pair of images related by a rigid transformation a point in one image can potentially match in the second image any point that lies along a straight line (which is referred to as "the *epipolar line* of that point").

Let P_1 and P_2 be two projections (either orthographic or perspective) of a rigid object from two given viewpoints. Let (x, y) be the projection of some object point in P_1.

Proposition 1: The corresponding point to (x, y) in P_2 lies along a straight line given by:

$$(x', y') = \mathrm{u} + \alpha(z)\mathrm{v}$$

where $\mathrm{u}, \mathrm{v} \in \mathcal{R}^2$ are constants (namely, independent of z), and α is a scalar function of z.

Following Proposition 1, given the transformation that relates the two images, the correspondence is determined up to a straight line. The vectors u and v are determined both by the transformation and by the 2-D position of the point (x, y), while α is the only component that depends on z, the depth value of the point in 3-D. There is a one-to-one mapping between the position of p along the epipolar line in P_2 and its depth value. Every different depth value corresponds to a different location of p along the epipolar line, and every different location along the epipolar line determines a different depth value.

In some cases point correspondence can be uniquely determined. This occurs in the degenerated case when $\mathrm{v} = 0$. In this case the position (x', y') does not depend on the depth value of the point. Under orthographic projection this occurs when the object is rotated around the line of sight and then translated arbitrarily. Under perspective projection v vanishes when the object is rotated around the camera.

The epipolar lines are parallel in the orthographic case, since in this case v depends solely on the transformation and is therefore common to all object points. This is not always true in the perspective case. In this case the epipolar lines are parallel only if the transformation includes no translation in depth. If, however, $t_z \neq 0$ the epipolar lines coincide at a single point known as the *focus of expansion*.

A rigid transformation divides the image into epipolar lines within which correspondence cannot be determined. Every epipolar line in one image has its corresponding epipolar line in the second image, in the sense that, all the points that lie along some epipolar line in the first image share the same epipolar line in the second image and vice versa. This is established in Proposition 2.

Proposition 2: Let $\mathrm{p}_1, \mathrm{p}_2 \in P_1$ be two points that lie along some common epipolar line. The epipolar line of p_1 and the epipolar line of p_2 in P_2 coincide.

Since rigidity alone does not determine the correspondence but up to epipolar lines, it may in some cases be sufficient to recover the epipolar lines rather than all the parameters of the transformation. Interestingly, under orthographic projection the epipolar lines are determined from two images while the transformation is not. Four non coplanar points are required for this task. The transformation breaks up into its planar parts and non planar parts. The planar parts of the transformation are determined by the epipolar lines, while the non planar parts, the rotation in depth, cannot be recovered. In the

perspective case both the epipolar lines and the transformation are determined from two images. In this case seven points are required.

The results above apply in two additional cases that extend beyond the set of rigid transformations. Epipolar lines exist when the objects considered undergo general 3-D affine transformation, which includes stretch and shear. The same applies under orthographic projection to objects with smooth bounding surfaces. In this case the contours change their position on the object with the viewpoint. (See a discussion in [Basri & Ullman 1988].) This motion is projected along epipolar lines (See section 2.3 below). In both cases, corresponding points lie along epipolar lines, and these epipolar lines can be recovered from a small set of corresponding points.

2.1 Orthographic Projection

In this section we repeat the results presented in the beginning of Section 2 and prove them for the orthographic case. Let P_1 and P_2 be two images of a rigid object from two arbitrary viewpoints. Let $\mathrm{p} = (x, y, z)$ be an object point, its position in P_1 is given by (x, y), and its position in P_2 is given by (x', y') which is the orthographic projection of $sR\mathrm{p} + \mathrm{t}$, where s is a scale factor, R is a 3×3 rotation matrix, and t is a translation vector.

In the following analysis we assume that the transformation between the images (namely, s, R, and t) is known. We select a point (x, y) in the first image and compute its possible positions in the second image. We show that the set of these positions forms a straight line, and that the exact position along this line is determined by its depth value.

Proposition 1a: Given a rigid transformation defined by $\{s, R, \mathrm{t}\}$ and a point $(x, y) \in P_1$, its corresponding point in P_2 lies along the epipolar line given by:

$$(x', y') = \mathrm{u} + z\mathrm{v}$$

where $\mathrm{u}, \mathrm{v} \in \mathcal{R}^2$ are constants.

Proof: Since

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s(r_{11}x + r_{12}y + r_{13}z) + t_x \\ s(r_{21}x + r_{22}y + r_{23}z) + t_y \end{pmatrix}$$

We define

$$\mathrm{u} = \begin{pmatrix} sr_{11}x + sr_{12}y + t_x \\ sr_{21}x + sr_{22}y + t_y \end{pmatrix}$$

$$\mathrm{v} = \begin{pmatrix} sr_{13} \\ sr_{23} \end{pmatrix}$$

Notice that since the transformation is given, u and v are determined, and the corresponding point for (x, y) lies along the straight line $\mathrm{u} + z\mathrm{v}$. When the depth value, z, of the point is given, the location of the corresponding point along the line is determined, and vice versa, selecting a corresponding point along the line determines its depth value. When $\mathrm{v} = 0$ the epipolar line vanishes into a point. In this case the images are separated by a rotation about the line of sight (plus some arbitrary translation). For symmetry reasons we obtain the same results for points in the second image, namely, that their corresponding points in P_1 lie along straight lines.

The epipolar lines in each of the images are parallel. This follows the fact that v depends solely on the transformation, and therefore has a common value for all image points. All the points in P_1 that lie along a single epipolar line share the same epipolar line in P_2. This is established in the following Proposition.

Proposition 2a: Let $\mathrm{p}_1, \mathrm{p}_2 \in P_1$ be two points that lie along some common epipolar line. The epipolar line of p_1 and the epipolar line of p_2 in P_2 coincide.

Proof: All the epipolar lines are parallel. According to the definition of an epipolar line, since p_1 and p_2 lie along a single epipolar line, both are possible matches of a single point, q, in P_2. Therefore, the epipolar lines of p_1 and p_2 intersect in q, and since epipolar lines are parallel they must coincide. Consequently, rigidity determines the correspondence between epipolar lines, but does not resolve the correspondence within these lines.

When only two images are given the transformation cannot be fully recovered. The epipolar lines, however, can be recovered using a correspondence set of four non coplanar points. A linear equation from which the epipolar lines can be computed is given below. We shall use the following notation. Let $(x_i, y_i) \in P_1$ and $(x'_i, y'_i) \in P_2$ be a pair of corresponding points, namely, they are the projections of a common point in 3-D space, $p_i = (x_i, y_i, z_i)$. We shall have n such correspondences. (To solve this equation n must be ≥ 4.) Denote $\mathbf{x} = (x_1, ..., x_n)$, $\mathbf{y} = (y_1, ..., y_n)$, $\mathbf{z} = (z_1, ..., z_n)$, $\mathbf{x}' = (x'_1, ..., x'_n)$, $\mathbf{y}' = (y'_1, ..., y'_n)$, and $\mathbf{1} = (1, ..., 1) \in \mathcal{R}^n$. According to [Ullman & Basri 1991], $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, $\mathbf{x}'$, $\mathbf{y}'$, and $\mathbf{1}$ are all embedded in a 4-D linear space. This follows the identities below

$$\begin{aligned} \mathbf{x}' &= sr_{11}\mathbf{x} + sr_{12}\mathbf{y} + sr_{13}\mathbf{z} + t_x\mathbf{1} \\ \mathbf{y}' &= sr_{21}\mathbf{x} + sr_{22}\mathbf{y} + sr_{23}\mathbf{z} + t_y\mathbf{1} \end{aligned}$$

Consequently, $\{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{1}\}$ span a 4-D linear space to which $\mathbf{x}'$ and $\mathbf{y}'$ also belong. Therefore, there exist nonzero scalars a_1, a_2, b_1, b_2, and c such that:

$$a_1\mathbf{x} + a_2\mathbf{y} + b_1\mathbf{x}' + b_2\mathbf{y}' + c\mathbf{1} = 0$$

These coefficients are determined (up to a scale factor) by four non coplanar points. The epipolar line are immediately derived from this equation. (This result is proved somewhat differently in [Huang & Lee 1989, Lee & Huang 1990].)

The epipolar lines break the transformation that relates the images into its planar components and its non planar ones. The planar components can be recovered from the epipolar lines, while the non planar ones cannot be determined from two images. The translation component perpendicular to the epipolar line is given by c. (The translation components can be discarded altogether if we consider differences between points rather than the points themselves.) The values of the other coefficients are given below.

$$\begin{aligned} a_1 &= sr_{32} & b_1 &= r_{23} \\ a_2 &= -sr_{31} & b_2 &= -r_{13} \end{aligned}$$

The scale factor is therefore given by the ratio

$$s = \sqrt{\frac{a_1^2 + a_2^2}{b_1^2 + b_2^2}}$$

The relative angle between the epipolar lines determines the planar parts of the rotation, as explained below. A 3-D rotation can be decomposed into a sequence of three successive rotations. A rotation about the Z-axis by an angle α, a second rotation about the Y-axis by an angle β, and a third rotation about the Z-axis by an angle γ. Under this decomposition the following identities hold

$$\begin{aligned} r_{32} &= \sin\alpha\sin\beta & r_{23} &= \sin\beta\sin\gamma \\ r_{31} &= -\cos\alpha\sin\beta & r_{13} &= \sin\beta\cos\gamma \end{aligned}$$

We therefore obtain that

$$\begin{aligned} \alpha &= \tan^{-1}\frac{a_1}{a_2} \\ \gamma &= -\tan^{-1}-\frac{b_1}{b_2} \end{aligned}$$

While β cannot be determined.

We can visualize this decomposition in the following way. After compensating for the translation and scale changes, we first rotate the image P_1 by α. Consequently, the epipolar lines point in P_1 to a horizontal direction. We then rotate the second image, P_2, by $-\gamma$. As a result, the epipolar lines in P_2 also point horizontally. The images obtained are related by a rotation about the vertical axis, which is a rotation in depth. Following such a rotation the points move horizontally, which is, along the (rotated) epipolar lines. This motion cannot be recovered since it depends both on the angle of rotation, β, and on the depth of the points.

2.2 Perspective Projection

In this section we repeat the results presented in the beginning of Section 2 and prove them for the perspective case. We use the following notation. An object point p is denoted by $z(x, y, 1)$. It is projected in P_1 to the position (x, y) and in P_2 to (x', y'). (There the actual 3-D position of the point is denoted by $z'(x', y', 1)$.)

Proposition 1b: Given a rigid transformation that includes a rotation R and a translation t, and given a point $(x, y) \in P_1$, its corresponding point in P_2 lies along the epipolar line given by

$$(x', y') = \mathrm{u} + \alpha(z)\mathrm{v}$$

where $\mathrm{u}, \mathrm{v} \in \mathcal{R}^2$ are constants, and $\alpha(z)$ is scalar.

Proof: Denote

$$\begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix} = R \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Note that x_r, y_r, and z_r are all independent of z. Since

$$z' \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = Rz \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} + \mathrm{t} = z \begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix} + \mathrm{t}$$

We obtain that

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \frac{1}{z_r} \begin{pmatrix} x_r \\ y_r \end{pmatrix} + \frac{1}{z'} \begin{pmatrix} t_x - t_z\frac{x_r}{z_r} \\ t_y - t_z\frac{y_r}{z_r} \end{pmatrix}$$

And so we define

$$\begin{aligned} \mathrm{u} &= \frac{1}{z_r} \begin{pmatrix} x_r \\ y_r \end{pmatrix} \\ \mathrm{v} &= \begin{pmatrix} t_x - t_z\frac{x_r}{z_r} \\ t_y - t_z\frac{y_r}{z_r} \end{pmatrix} \\ \alpha(z) &= \frac{1}{z'} = \frac{1}{zz_r + t_z} \end{aligned}$$

Parallel epipolar lines are obtained when $t_z = 0$. In this case v is independent of the position of the point and depends solely on the transformation. If, however, $t_z \neq 0$ the epipolar lines intersect in one point, called the *focus of expansion*. This point stands for $z = 0$, and its location in P_2 is given by

$$\begin{pmatrix} x'_0 \\ y'_0 \end{pmatrix} = \frac{1}{t_z} \begin{pmatrix} t_x \\ t_z \end{pmatrix}$$

The location of the focus of expansion in P_1 corresponds to the case when $\mathrm{v} = 0$. This condition implies the following linear equation system

$$\begin{aligned} t_z x_r &= t_x z_r \\ t_z y_r &= t_y z_r \end{aligned}$$

From which this location can be retrieved. (Recall that x_r, y_r, and z_r are linear functions of x and y.)

Similar to the orthographic case, points that lie on a common epipolar line in one image share the same epipolar line in the other.

Proposition 2b: Let $\mathrm{p}_1, \mathrm{p}_2 \in P_1$ be two points that lie along some common epipolar line, assume both p_1 and p_2 are not the focus of expansion. The epipolar line of p_1 and the epipolar line of p_2 in P_2 coincide.

Proof: If $t_z = 0$ the epipolar lines are parallel and the proof is identical to that of the orthographic case. If $t_z \neq 0$ the epipolar lines in each image intersect in the focus of expansion. Since the points lie along a common epipolar line in P_1 there exists a point q in P_2 that is a possible match to both points, q is not the focus of expansion. Therefore, the epipolar line of p_1 and that of p_2 intersect in q, and since both lines also intersect in the focus of expansion they must coincide.

In the perspective case the transformation can be determined (up to a scale factor) by a correspondence set of seven points [Longuet-Higgins 1981, Tsai & Huang 1984]. For the sake of completeness we review in Appendix A one method to recover the transformation from eight corresponding points using essentially linear operations. This method appeared in Tsai & Huang [1984].

It is worth noting that although in the perspective case the transformation can be recovered from two images the computation may in many cases be unstable. This happens when the object is relatively distant from the camera, in which case depth differences are relatively small and perspective distortions are negligible, and when the depth translation component vanishes, in which case the epipolar lines are parallel. These cases are essentially similar to the orthographic case. In both cases the transformation obtained is unstable, but the epipolar lines are still stable.

2.3 Extensions

In the previous discussion we showed that rigidity determines the correspondence up to epipolar lines and that the position of points along these lines is determined by

their depth values. We also showed that the epipolar lines can be recovered from a small set of corresponding points. In this section we extend these results to two additional cases, images of objects that undergo general affine transformation and contour images of rigid objects with smooth bounding surfaces.

An affine transformation in 3-D space is composed of a general linear transformation followed by a translation. The set of affine transformations contains in addition to all the rigid transformations also stretch and shear. Similar to the rigid case, in a pair of images of an object that undergoes an affine transformation, corresponding points lie along epipolar lines. This is true both under orthographic as well as perspective projections. It follows the fact that in proving the results above we never used the special properties of the rotation matrix.

When a pair of images is given, whether the objects in these images are moving rigidly or whether they undergo an affine (non rigid) transformations is indistinguishable. Basri & Ullman [1991] showed that under orthographic projection the set of images of a rigid object is contained in a 4-D linear space, and additional (quadratic) constraints distinguish between these images and other vectors in this space. These other vectors are, in fact, images obtained by applying a general 3-D affine transformation to the object. The quadratic constraints cannot be recovered from two images. Hence, it is impossible to distinguish between the two cases when only two images are given. A similar ambiguity holds under perspective projection. It is worth noting that general affine transformations approximate the way moving objects are observed in movies from different viewpoints. This effect is known since 1859 as the La Gournerie Paradox and was discussed recently by Jacobs [1991].

A second interesting case is that of rigid objects with smooth surfaces. The bounding contours of such an object are generated by surface patches that are tangent to the line of sight. These patches are usually referred to as the *rim* [Koenderink & Van Doorn 1979] or the *contour generator* [Marr 1977] of the object. Since the surface of the object is smooth, when the object rotates in depth a new set of surface patches that are now tangent to the new line of sight replaces the original rim, generating a new set of bounding contours. Establishing correspondence between the original and the new bounding contours of the object is therefore problematic, since the contours undergo in addition to the rigid transformation also some arbitrary motion that depends on the exact shape of the object.

Tracing the positions of these contours is useful for any shape reconstruction and object recognition scheme that is based on contour matching. A method to predict the appearance of objects with smooth surfaces for recognition was recently developed [Basri & Ullman 1988]. The method assumes an orthographic projection and uses the 3-D curvature of points along the contours to follow their change in position with viewpoint. The curvature values were computed from a few images of the object by matching the contours in these images.

The next observation demonstrates that in the case of objects with smooth surfaces under orthographic projection corresponding points lie along epipolar lines. We first look at the simpler case of an object that rotates about the vertical axis. Let p be a rim point, and let us take a horizontal slice of the object that contains p. (Namely, if p $= (x_0, y_0, z_0)$ we consider the plane $y = y_0$.) The intersection of the surface of the object with this plane forms a space curve, C. When the object rotates, the rim point p changes its position on the object along C. Denote the new rim point by p'. Since this is a rotation about the Y-axis, the epipolar lines in both images are horizontal. Therefore, all the points on C including p and p' are projected to a common epipolar line in both images.

We now extend this observation to general rigid transformations. Rotation is the only component that affects the rim. Translation and scaling do not change the rim and therefore can be disregarded. A 3-D rotation can be decomposed into three successive rotations, around the Z-, Y-, and Z-axes. (The same decomposition used in Section 2.1.) As we did in Section 2.1, we apply the first rotation to the first image, and (the inverse of) the last rotation to the second image. Both rotations are image rotations, and they do not change the rim. After applying these rotations we obtain that the two images are related by a rotation about the vertical axis, and hence their epipolar lines are horizontal. (See Section 2.1.) Therefore, the observed position of the rim points change along epipolar lines.

Figure 1 shows the epipolar lines in two orthographic projections of a VW car. Notice that the matching between silhouette points along epipolar lines is good although they are generated by smooth surfaces.

3 Resolving Point Correspondence

In the previous section we have shown that rigidity alone is insufficient to solve the correspondence problem uniquely from two images. It divides the images into epipolar lines, their matching is determined by the transformation that separates the images, but the correspondence of points within the lines cannot be resolved. In this section we examine the problem of establishing correspondence in three or more images. We show that, as in the case of two images, the correspondence is not determined uniquely. Additional images, however, provide constraints that can be used to solve the problem heuristically (e.g., the trinocular stereovision algorithm [Yachida 1986]). We discuss several additional constraints that can be used together with epipolar lines to find the correspondence between images. These methods were implemented and the results are presented below.

It should be noted that the use of epipolar lines to determine correspondence is limited to those regions in the images that are consistent with a rigid (or affine) transformation. When the images contain a number of rigid objects moving independently each of the objects may determine a different set of epipolar lines. A segmentation process must be applied to separate these objects and divide the images into regions with consistent sets of epipolar lines. We shall not address the segmentation problem in this paper.

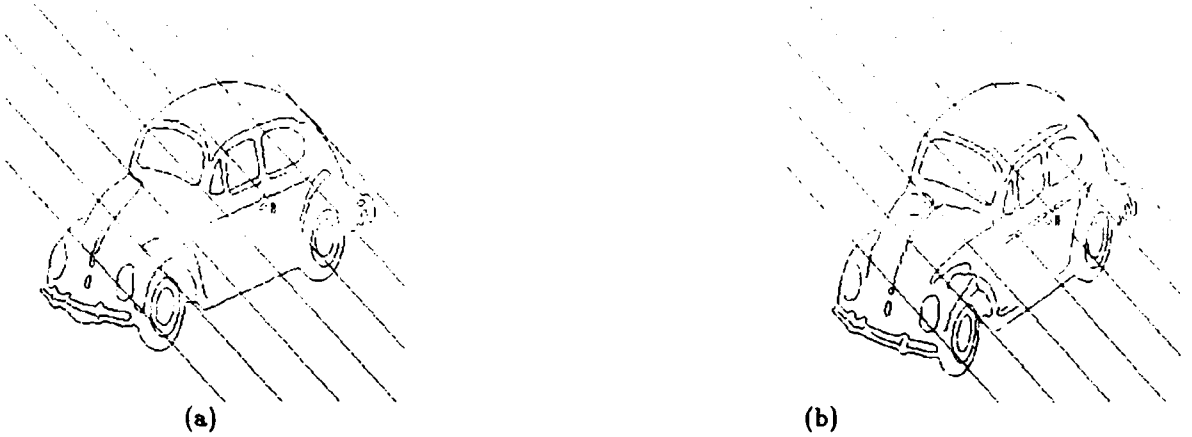


Figure 1: Epipolar lines in two orthographic projections of a VW car. Note the fact that corresponding points lie along epipolar lines. The silhouette contours deserve special attention for being generated from smooth surfaces.

3.1 Correspondence from Three Images

We have so far explored the establishment of correspondence from two images. We showed that the correspondence between points in the images cannot be uniquely resolved even if the transformation is known. We now address the following question. Can the correspondence be resolved when three or more images are considered? Structure from motion theory demonstrates that the answer to this question is not trivial. When correspondence is given, under orthographic projection two images are not sufficient to recover the transformation, but three are [Ullman 1979, Huang & Lee 1989]. The correspondence problem is nevertheless different from the structure from motion problem. Point correspondence cannot be resolved by using any number of additional images. Yet, additional images provide information that can be used to filter out less likely solutions.

Proposition 3 establishes that point correspondence cannot be resolved from any number of images. Let $P_1, P_2, ..., P_k$ be k images. Let (x_i, y_i), $1 \leq i \leq k$ be the locations of a point $p = (x, y, z)$ in P_i. (Assume w.l.g. that $x_1 = x$ and $y_1 = y$.) Let T_i, $2 \leq i \leq k$ be the rigid transformation applied to p in P_i, assuming orthographic projection.

Proposition 3: Given $T_2, ..., T_k$, the set of possible locations of p in $P_2, ..., P_k$ is a straight line in $\mathcal{R}^{(k-1)\times 2}$ given by:

$$(x_2, y_2, ..., x_k, y_k) = \mathbf{u} + z\mathbf{v}$$

where $\mathbf{u}, \mathbf{v} \in \mathcal{R}^{(k-1)\times 2}$ are constants.

Proof: This is obtained simply by defining $\mathbf{u} = (u_2, ..., u_k)$ and $\mathbf{v} = (v_2, ..., v_k)$, where $u_i, v_i \in \mathcal{R}^2$ are the corresponding vectors u and v from Proposition 1a.

This proposition implies that the number of possible correspondences for each point is infinite. Every possible assignment of z yields to a different location of the points in all of the images. An equivalent claim can be made in case of perspective projection.

There is, however, one additional consequence to this proposition. Determining the correspondence between two of the images immediately implies the correspondence in all other images. This property suggests a hypothesis-verification heuristic to recover correspondence. The algorithm first selects a point in the first image, hypothesizes its correspondence in the second image, computes accordingly its position in the third, and then verifies its appearance in the predicted location. This algorithm is used in Trinocular stereopsis [Yachida 1986]. The algorithm can be defined in two versions. The first requires the transformation between the images. It predicts the position of points in the third image by explicitly computing their depth values. The second requires the epipolar lines between all pairs of images. It predicts the position of points in the third image by intersecting epipolar lines.

Version 1.

1. Select a point $p = (x, y) \in P_1$ and find its epipolar lines A in P_2.
2. For all candidates $q_1, ..., q_n$ along A compute the corresponding depth value $z_1, ..., z_n$.
3. For every possible depth value, $z_1, ..., z_n$, compute the position of the point (x, y, z_i) in P_3 and verify its actual appearance at this location.

Version 2.

1. Select a point $p = (x, y) \in P_1$, and find its epipolar lines A in P_2 and B in P_3.
2. For all candidates $q_1, ..., q_n$ along A compute their epipolar lines $C_1, ..., C_n$ in P_3.
3. Intersect each of the lines, $C_1, ..., C_n$, with B and verify the actual appearance of p in these locations.

The two versions of the algorithm are essentially similar. The first version uses the transformation between the images to compute depth values. The second version replaces this computation by intersecting epipolar lines. Note that the transformation can be computed from three images using four non coplanar points [Ullman 1979]. The second version can be used only if the epipolar lines C_i intersect with B. The meaning of this requirement is for every image its epipolar lines with

respect to the other images should all be non parallel. This requirement is equivalent to requiring the transformations to be independent. Unless this condition is met structure-from-motion algorithms cannot recover the transformation from correspondence [Huang & Lee 1989].

One observation following this algorithm is that, since epipolar lines are defined for pairs of images, one can use different sets of anchor points to recover the epipolar lines in each of the pairs. This is different from most existing structure from motion algorithms, which require from the set of anchor points to be identical in all three images.

Note that the use of three images rather than two is reasonable since three images are required to recover structure from motion under orthographic projection [Ullman 1979] and to form a viewer-centered representation for a rigid object [Ullman & Basri 1991].

The algorithm handles both rigid objects as well as objects that undergo general 3-D affine transformations. There is, however, some difference between the two cases. When four or more images are considered certain configurations of epipolar lines may be consistent with some affine transformations but with no rigid ones. This is concluded from [Basri & Ullman 1991], since three images are necessary to determine the functional constraints that distinguish rigid transformations from affine ones. These constraints then restrict the possible configuration of the epipolar lines in larger sets of images.

It should be stressed that both versions do not guarantee uniqueness. From time to time several candidates may be found consistent with all three images. Further pruning between these candidates is required. In general the algorithm gives better results for sparse images than for dense ones and for images with arbitrarily distributed texture than for images with uniform texture. (Density refers here to the number of points actually considered by the algorithm relative to the total area of the images.) A common way to reduce the density of an image is to consider its edge map. Edge images are in general still too dense, and a naive implementation of the algorithm would fail to provide a unique solution for many of the points. To avoid this problem we suggest to apply this matching procedure to edges rather than to points, using the assumption that continuous edges tend to remain continuous in all images. Unlike Ayache & Lustman [1987], our implementation is not confined to straight line segments, but is applied to arbitrarily curved ones. We exploit the shape variance of image contours to discriminate between correct and false matches.

The modified algorithm was implemented and run on natural images. An example is given in Figures 2-4. In these figures correspondence was sought between three edge images of a VW car (Figure 2). We first selected a contour from the first image. Then we found all the contours in the second image that could possibly match the selected contour. For each of the candidates we computed their location in the third image. We repeated this process for a number of contours. Figure 3 shows the best candidates projected to the third image. Figure 4 shows some of the other candidates projected. None of these candidates match an actual contour (although some of their points do). The results of this algorithm were used to create object models for recognition. An example for the use of these models can be found in [Ullman & Basri 1991].

3.2 Alternative constraints

In this section we briefly discuss several constraints that, combined with the epipolar lines, are useful in establishing point correspondence. The first constraint is traditionally referred to as *orderness*. Most objects are opaque. Contour segments (and points) on such objects retain their spatial order from different viewpoints. Therefore, a contour segment B that lies between two segments, A and C, in one image would in general match some contour segment B′, which lies between the two corresponding segments, A′ and C′ respectively. (Notice that right, left, up, and down can still change, as in the case of a 180° rotation around the line of sight.)

Other cues that may be helpful to resolve the correspondence are parallelism and symmetry. If a pair of contour segments are parallel or symmetrical in one image their corresponding segments in the second image are often parallel or symmetrical respectively. Resolving the correspondence for one segment would therefore indicate a solution for the other segment. It is worth mentioning, however, that perspective projection does not maintain parallelism, and that symmetrical components often appear skewed in the image under both projections. Incorporating these cues into a process of resolving the correspondence may therefore be fairly difficult.

Epipolar lines can be used to improve correspondence achieved under aperture conditions. Under these terms matching between contours is given along a direction perpendicular to the contours [Marr & Ullman 1981]. Common techniques to correct the matching use iterative computation to maximise the smoothness of the flow [Hildreth 1984], or use sequences of images to find a rigidly consistent solution [Ullman 1984]. The epipolar line technique offers an exact solution to the aperture problem for rigid motion that is both computationally simple and resolves the correspondence for as few as two images.

Figure 5 compares the matching obtained under aperture conditions with the matching obtained using epipolar lines for two car silhouettes. It should be noted that in general the aperture problem is associated with short range motion applications. In this case the computation of epipolar lines tends to be unstable. One way to overcome this problem is to recover the epipolar lines for a sequence of images, such that the difference between each pair of consecutive images is small, but the overall transformation accumulated along the sequence is large. Alternatively, if two "distant" images are provided the images may first be roughly aligned before aperture matching can take place.

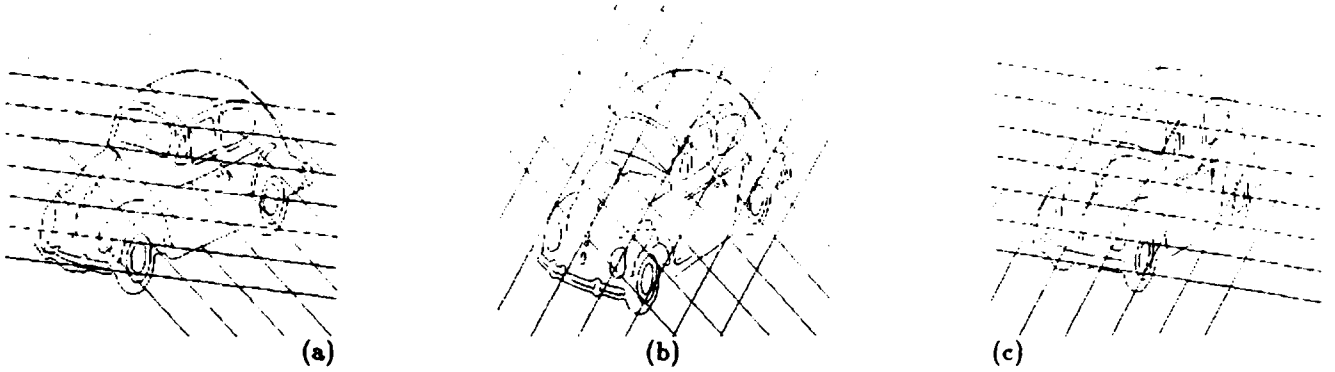


Figure 2: Epipolar lines in three images of a VW car. Every image contains one set of epipolar lines against each of the other two images.



Figure 3: Application of the three images algorithm to four contour pieces selected from the car in Figure 2(a). (The selected contours include the roof silhouette, the front window, the rear side window, and the bottom silhouette.) (a) The best prediction found by the algorithm for the four contour pieces. (b) This prediction overlapped with the actual (third) image.

4 Summary

The recovery of shape from a motion sequence requires in general establishing correspondence between the points in the images. This task is particularly difficult when the images are taken from viewpoints that are relatively distant from one another, conditions referred to as "long range motion". Establishing correspondence under these conditions is important for building both object-centered as well as viewer-centered representations for object recognition. Such representations tend to be more stable as the images from which they are constructed are separated by relatively large transformations.

Information about the shape of objects and the transformations they undergo can be used to guide the matching process. In this paper we reviewed the constraints imposed on the correspondence by rigid transformations and extended them to include images of objects that undergo general 3-D affine transformations as well as rigid objects with smooth surfaces. In all these cases the images are divided into epipolar lines, their correspondence is determined by the transformation, but the correspondence of points within the lines cannot be recovered. The epipolar lines can be computed from a small set of anchor points.

The correspondence is not determined uniquely even wh n three or more images are considered. Additional im ges can be used, however, in a heuristic algorithm to d etermine point correspondence. Such an algorithm is the trinocular stereovision algorithm [Yachida 1986], which is designed to work with sparse images and in the absence of uniform texture. We extended this algorithm to handle arbitrarily curved edge images and applied it to images of natural objects. We discussed the use of other constraints such as orderness, parallelism, and symmetry in solving the correspondence problem. Finally, we showed that epipolar lines can be used to improve matching obtained under aperture conditions. The techniques described in this paper were implemented and used to construct viewer-centered models for object recognition.

Acknowledgments I wish to thank Shimon Ullman, Eric Grimson, Tao Alter, Amnon Shashua, and Yael Moses for their helpful comments throughout the work. This research was sponsored in part by the McDonnell-Pew Foundation. This report describes research done at the Artificial Intelligence Laboratory of the Mas-

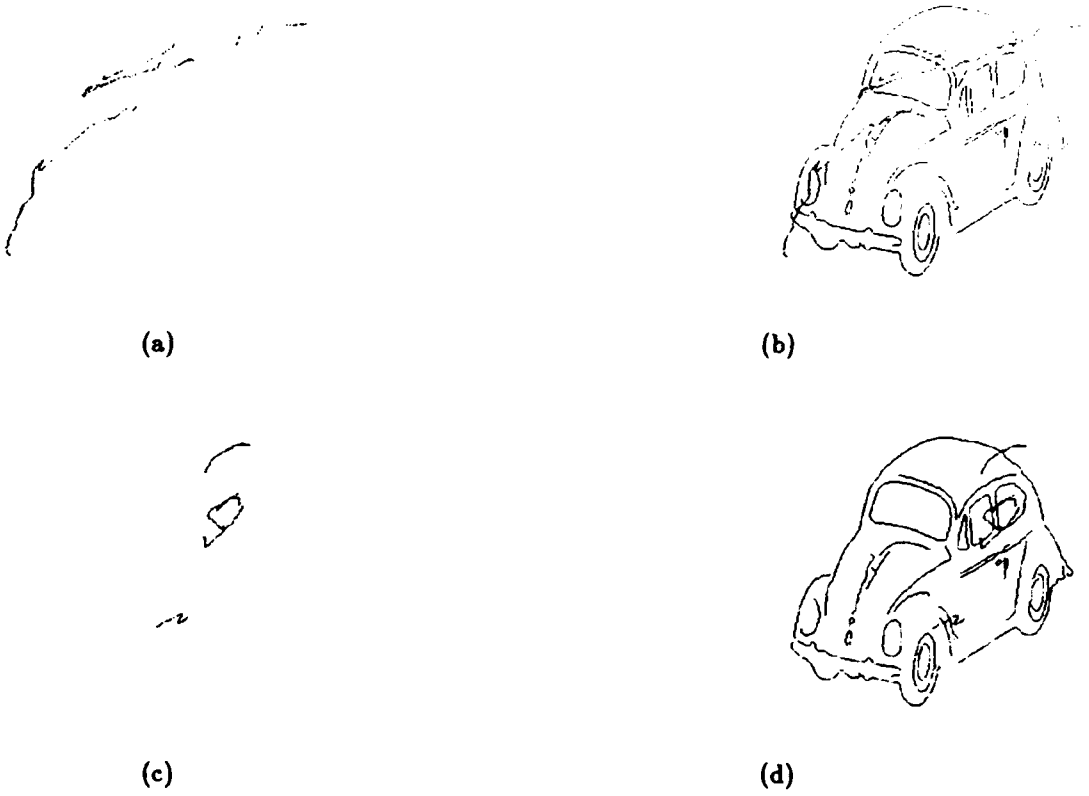


Figure 4: Correspondence candidates that were not selected by the algorithm because their predictions poorly matched the third image. (a) Prediction of false candidates. (b) This prediction overlapped with the actual image. (c) Another prediction of false candidates. (d) This prediction overlapped with the actual image.

sachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by National Science Foundation, contract number IRI 8900267, and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124.

References

Ayache, N. & Lustman, F., 1987. Fast and reliable passive trinocular stereovision. *Proc. of ICCV, London, UK*, 422-427.

Baker, H.H. & Binford, T.O., 1981. Depth from edges and intensity based stereo. *Proc. of IJCAI, Vancouver, BC*, 631-636.

Basri, R. & Ullman, S., 1988. The alignment of objects with smooth surfaces. *Proc. of 2nd ICCV, Tampa, FL*, 482-488.

Bolles, R.C. & Baker, H.H., 1985. Epipolar-plane image analysis: a technique for analyzing motion sequences. *3rd Int. Sym. on Rob. Res., Paris, France.*

Fischler, M.A. & Bolles, R.C., 1981. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Com. of the ACM 24(6)*, 381-395.

Grimson, W.E.L., 1980. A computer implementation of a theory of human stereo vision. *Phil. Trans. of the Royal Soc., London. B 292*, 217-253.

Grimson, W.E.L. & Lozano-Pérez, T., 1984. Model-based recognition and localization from sparse data. *Int. J. of Rob. Res., 3*, 3-35.

Grimson, W.E.L. & Lozano-Pérez T., 1987. Recognition of object families using parameterized models, *Proc. of ICCV*, 93-100.

Hildreth, E.C., 1984. Computations underlying the measurements of visual motion. *AI 23*, 309-354.

Huang, T.S. & Lee, C.H., 1989. Motion and Structure from Orthographic Projections. *IEEE Trans. on PAMI 2(5)*, 536-540.

Huttenlocher, D.P. & Ullman, S., 1987. Object recognition using alignment. *Proc. of ICCV, London, UK*, 102-111.

Jacobs, D.W., 1991. *To be published.*

Koenderink, J.J. & Van Doorn, A.J., 1975. Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer. *Optica Acta 22(9)*, 771-773.

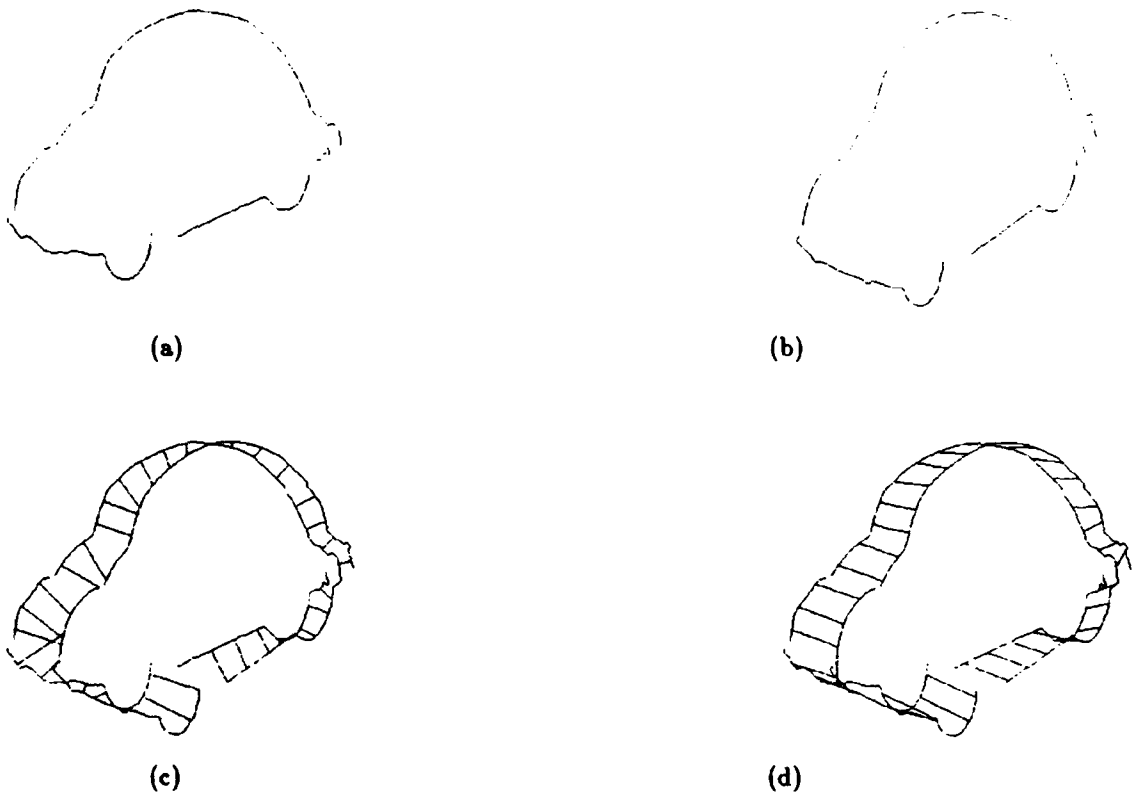


Figure 5: Matching silhouettes of a VW car under aperture conditions versus to using epipolar lines. (a,b) Two silhouette pictures of a VW car from two different viewpoints. (c) Matching the two silhouettes under aperture conditions. (d) Matching the two silhouettes by epipolar lines. (In these two figures the two silhouette drawings are overlapped. Straight lines connect matching points.)

Koenderink J.J. & Van Doorn A.J., 1979. The internal representation of solid shape with respect to vision. *Bio. Cyb. 32*, 211-216.

Lee, C.H. & Huang, T.S., 1990. Finding point correspondences and determining motion of a rigid object from two weak perspective views. *CVGIP 52*, 309-327.

Longuet-Higgins, H.C., 1981. A computer algorithm for reconstructing a scene from two projections. *Nature 293*, 133-135.

Lowe, D.G., 1987. Three-dimensional object recognition from single two dimensional images. *AI 31*, 355-395.

Marr D., 1977. Analysis of occluding contour. *Phil. Trans. of the Royal Soc., London, B 275*, 483-524.

Marr, D. & Poggio, T., 1979. A computational theory of human stereo vision. *Proc. of the Royal Soc., London. B 204*, 301-328.

Marr, D. & Ullman, U., 1981. Directional selectivity and its use in early visual processing. *Proc. of the Royal Soc., London. B 211*, 151-180.

Tsai, R.Y. & Huang, T.S., 1984. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Trans. on PAMI 6(1)*, 13-27.

Ullman, S., 1978. Two dimensionality of the correspondence process in apparent motion. *Perception 7*, 683-693.

Ullman, S., 1979. The Interpretation of visual motion. *MIT Press, Cambridge, MA.*

Ullman, S., 1983. Recent computational studies in the interpretation of structure from motion. In: A. Rosenfeld & J. Beck (Eds.), *Human and Machine Vision. Academic Press, NY, NY.*

Ullman, S., 1984. Maximizing rigidity: the incremental recovery of 3-D structure from rigid and rubbery motion. *Perception 13*, 255-274.

Ullman, S. & Basri, R., 1991. Recognition by linear combinations of models. *IEEE Trans. on PAMI 13(10).*

Yachida M., 1986. 3-D data acquisition by multiple views. *3rd Int. Sym. on Rob. Res., MIT Press, Cambridge, MA*, 11-18.

Iterative TIN Generation from Digital Elevation Models

Michael F. Polis, David M. McKeown, Jr.

Digital Mapping Laboratory
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

In this paper we describe a new technique to produce a triangulated irregular network (TIN) from a digital elevation model (DEM). The overall goal is to produce an approximate terrain description that preserves the major topographic features using a greatly reduced set of points selected from the original DEM. The TIN generation process is iterative; at each iteration we identify areas in the DEM that lie outside of a user-supplied error tolerance in the TIN, and choose points from the DEM to more accurately model these areas. Point selection involves the computation of the difference between the actual DEM and an approximate DEM. This approximate DEM is calculated by interpolating elevation points from the TIN.

The iterative nature of the algorithm permits users to terminate the terrain approximation algorithm based on operational criteria, such as median error, maximal error, or number of points used to construct the TIN. This is particularly relevant to real-time computer image generation as various scene rendering systems utilizing polygonal terrain patches have well defined limitations in order to maintain real-time performance. It also implies that the TIN generation procedure is automatically sensitive to smooth or rough terrain in that it selects only enough points to satisfy the required error constraints and tends to place points in those areas having the greatest topographic complexity.[1]

[1]This research was primarily sponsored by the U.S. Army Topographic Engineering Center under Contracts DACA72-87-C-0001 and DACA76-91-C-0014 and partially supported by the Avionics Lab, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Topographic Engineering Center, or the Defense Advanced Research Projects Agency, or of the United States Government.

1. Introduction

The most common format for storing terrain data is in a regular rectangular array of points known as a digital elevation model (DEM). DEM's can be generated using a number of methods, including direct point measurement and by interpolation from cartographic source material. In the case of point measurement, data can be compiled using stereo imagery where elevation points are measured on a regular grid, by collecting along lines using a manual analytic stereo plotter, or using automatic stereo correlation. DEM's can also be created by digitizing contour maps and interpolating elevations on a regular grid. The DEM is an easily generated and commonly available format, but a regular array of points is not an efficient representation in terms of information for the number of points used.

An alternative terrain representation format is a triangulated irregular network (TIN), consisting of a set of points freely placed in 3 dimensions and connected in a manner, nearly always planar, so as to approximate a surface by triangular patches. TIN's can be used to model surfaces whose elevations or properties are not easily sampled, such as aquifers or seismic data. The use of TIN's which concerns us, however, is the representation of a surface which is easily sampled at arbitrary locations. This is the case, for example, when a DEM is available which is of higher resolution than the desired end product.

TIN's are a more efficient model than DEM's for processing information in a variety of problems. They provide an advantage in many applications simply by reducing the sheer volume of data which must be considered. A TIN is capable of greater efficiency because it can adapt its resolution to the complexity of the terrain. Terrain display is a task for which TIN's are well-suited, especially in the case of real time applications. When determining which parts of a scene are visible, a TIN presents a relatively small number of non-overlapping polygons (all triangles) which represent the terrain, simplifying graphics calculations. Shading and texture-mapping these triangular

representations are also straightforward. TIN's are also useful in terrain analysis, as they represent terrain which, in many cases, has already been partially analyzed. Peaks, valleys, and ridge and channel networks are all easily found as they correspond to vertices and edges in the TIN. Given a description of the topography in terms of these features augmented by slope and aspect information, the extraction of drainage patterns and water sheds is simplified.

In this paper we describe a new technique to produce a triangulated irregular network (TIN) from a digital elevation model (DEM). The overall goal is to produce an approximation of the terrain description that preserves the major topographic features using a small subset of points selected from the original DEM. The TIN generation process is iterative; at each iteration we identify areas in the DEM that lie outside of a user-supplied error tolerance in the TIN, and attempt to choose points in the DEM to more accurately model these areas. Point selection involves the computation of the difference between the actual DEM and an approximate DEM. This approximate DEM is calculated by interpolating elevation points from the TIN.

In Section 2 we describe the overall structure of our TIN generation program based upon iterative refinement and our methods for point selection and error evaluation. In Section 3 two examples are presented to illustrate the TIN generation process using real DEM's in complex rugged terrain. In the following section we provide some background in this research area.

1.1. Previous Work

Interest in the generation and utilization of irregular terrain representations dates back nearly twenty years. The work of Peucker [Peucker, et. al. 76, Peucker, et. al. 78] introduced the TIN terminology and outlined the basic TIN construction problem in terms of sampling constraints, as well as manual and automated techniques for triangulation. They also described the importance of topological structure in selecting points for inclusion in the TIN. Over the years a variety of techniques for manual and automatic point selection have been proposed and implemented.

Traditionally, points have been selected from stereo imagery by a skilled operator [Carter 88]. Recently, work has been done on automatic TIN generation from other terrain models already rendered in digital form. For example, Chen and Guevara [Chen and Guevara 87] generate a TIN from raster data, and Christensen [Christensen 87] generates a TIN from contours on a digitized topographic map. In early work, contemporaneous with Peucker, Mark [Mark 75] reported that a TIN of approximately 14 times fewer points could be constructed to achieve the same accuracy as the original DEM. This result was confirmed by Peucker, and appears to have been generally accepted.

Triangulation can be performed manually, but Peucker [Peucker, et. al. 76] noted that automatic triangulation exhibits comparable performance. Given automatic triangulation, TIN generation techniques can be broadly divided into two categories: those which use manual point selection, and those which use automatic point selection. We survey some of the more recent published results.

One of the earliest manual point selection methods is the one used in the ADAPT system. This system, mentioned in Peucker [Peucker, et. al. 76], uses man-machine interaction to aid the user in producing a TIN which accurately represents the original terrain. The system goes through an iteration process in which it indicates inadequate parts of the TIN, which the human operator then improves.

Christensen [Christensen 87] uses TIN's to interpolate between digitized contours. The input contour is represented as a polygon, with the next contour, if there is one, indicating an area or areas cut out of the middle of the present contour. A medial axis of this shape is computed, and then a Delaunay triangulation is performed inside the shape, using the vertices of the contour and medial axis as the points to be triangulated.

The work most closely related to ours is that by Chen and Guevara [Chen and Guevara 87]. Their VIP procedure for selecting points directly from the DEM is based on the distance a point is from the 4 lines connecting its neighbors. This would seem to be a good solution to the point selection problem; all points in the DEM are ordered in terms of importance, which permits selection of a point set of any given size. The procedure does have several problems, however, all of which stem from the local nature of the importance criterion. First, the peak of a small, sharp hill will be considered more significant than the peak of one which is large, yet slopes gently. In the paper, this problem appears to have been circumvented by choosing enough points so that ridges and valleys appear clearly. Second, the VIP procedure chooses nearly all points on ridges and valleys. This is desirable in the sense that these are important features to capture. However, this is clearly unnecessary in those places where the ridge or valley follows a straight line, since we can represent straight lines with two points. Both of these problems arise because only the 8 neighboring points of any point are considered. In fact, for good point selection, we need to consider topographic features on a more global basis. Yet, the more points we consider, the more computationally intensive the procedure becomes. This suggests that a less direct method for point selection might have better performance.

Our algorithm, like the one used in the ADAPT system, is iterative, choosing points in stages until the

overall quality is satisfactory. It differs from the ADAPT system in that it does not require a human operator to select points; like the VIP algorithm, it chooses points automatically. But, unlike the VIP algorithm, the significance of each point is not considered independently of surrounding points. Since we have structured the algorithm to provide a previous approximation, we find regions in which there are currently errors and choose a few points to correct each.

2. Iterative TIN Generation

There are two main issues for TIN generation given an existing DEM: point selection and triangulation. The point selection problem can be stated as: "Given some maximum number of points, which should be chosen so as to best represent the terrain?" For triangulation the issue is: "How can we best connect a given set of points to represent the terrain?" Clearly answers to these two questions are interdependent. This section describes our iterative point selection algorithm and modifications to standard planar triangulation in some detail.

Each iteration of the algorithm adds a variable number of corrective points (selected from the DEM) in regions where an error term has a local maximum. Clearly, points in these areas will be necessary if the TIN is to be improved. The algorithm terminates when there no longer exist any error regions whose size is greater than a minimum area threshold, and whose error values are above a given error threshold ε. Varying ε allows loose control over the quality of the resulting TIN's. The algorithm goes through the following sequence of steps:

1. Initialization.
2. Generate the difference DEM by computing the absolute value of the difference between the actual DEM and the interpolated DEM approximation at each point in the DEM.
3. Select contour intervals and compute error contours from the difference DEM.
4. Select representative points based upon error contours.
5. Filter points based upon adjacency criterion.
6. Add points to the set of selected points and reconstruct the TIN.
7. Interpolate an approximate DEM from the TIN.
8. Evaluate the quality of the approximate DEM with respect to error regions and point selection budget. If the error is exceeded, and points are still available, return to Step 2.

2.1. Initialization

We start by selecting the corner points of the DEM as the initial set of points for the TIN; this will ensure that the boundaries of the TIN and DEM will be identical, enabling us to evaluate the error at each point in the TIN. The 4 corner points are triangulated, as described in Section 2.3, to produce an initial TIN. A DEM is generated from the TIN by linear interpolation of each triangle in the TIN, and the absolute value of the difference between the interpolated TIN and the DEM at each point becomes the initial difference image. We then compute the histogram of the difference image, and use it to find the maximum error δ, defined as the highest point on the histogram which has more than *minarea* pixels. We do not use the absolute maximum, as we are willing to tolerate small areas of relatively high error as long as the majority of the surface is well represented.

2.2. Point selection and filtering

For preliminary point selection we first seek points representative of the error regions found in the difference image, for these identify regions in the DEM which we are not modeling adequately. We determine the areas in which we need points by contouring the difference image. The algorithm tries to generate 4 contours equally spaced in the error range, i.e. at $\delta/4$, $\delta/2$, $3\delta/4$ and δ. In this way, we can correct local error maxima as well as global error maxima. As δ gets small, the position and number of contours are adjusted to obtain a reasonable contouring which does not include error regions with errors below ε, and which does not include error regions of negligible size. By doing this, we ensure that our error regions have both a sufficiently large error and coverage to warrant further consideration. We meet the coverage constraint by limiting the minimum distance between contours to 2, and we meet the error constraint by limiting the lowest contour to ε-2. If we were to limit the lowest contour to ε, we would inhibit the algorithm's ability to capture error regions which have more structure just below ε.

The next step is to filter the output of the contouring program to obtain error region shapes. These polygons must be sufficiently large to allow a meaningful medial axis to be obtained, which we motivate shortly. The size is controlled by a minimum area limitation, *minarea*, for the resulting contours. Then the polygons which pass the filter will be those which cover at least *minarea* pixels and which do not contain any contour of at least *minarea* pixels.

We then select points to represent these error polygons. One could simply take the centroid of each polygon, but this has two disadvantages: the centroid will not necessarily be in the error polygon, and merely using the centroid throws away information we have about the error region shape. Instead, a medial axis is found for each contour. Before computing the medial axis, the shape of the contour is smoothed by dilating and then eroding the interior of the contour by a factor of 3.

This serves to remove bumps from the contour that would otherwise cause the medial axis routine to produce a spur leading out to the bump. The medial axis transform is then applied to the filtered contour. Finally, the number of vertices in the medial axis is reduced by a smoothing routine [Aviad 88] which approximates the medial axis using a point at every intersection, every corner more than 135 degrees sharp, and every endpoint. The medial axis approximation has fewer vertices, allowing us to constrain the number of candidate points for the TIN, while still maintaining the basic structure of the error polygon.

The next step involves the selection of a set of these candidate points for addition to the TIN. These points are all representative of errors in the current approximation. Some of these points, however, may be very close to one another (consider the medial axis of an almost-square rectangle). We begin by using all the points from the previous approximation in the current approximation. This serves to drive the process toward convergence, although this strategy means that errors caused by a poor point choice are difficult to fix. Next we add points from the candidate set, one at a time, only adding a point if it is not too close to a point already in the current set of points. We judge points to be too close if they are within *mindist* pixels in both horizontal (x and y) directions and *minheight* (experimentally, $\varepsilon/2$ seems to be a good value for *minheight*) pixels in the vertical (z) direction. If this filtering process rejects all the candidate points, then all the points are added since we have no mechanism for choosing a point from a group of points in close proximity.

2.3. Triangulation

The new set of points is then triangulated to produce a new TIN; the TIN is interpolated to produce a DEM; and a new difference image is produced for the next iteration. Although the Delaunay triangulation is the most popular for generating a TIN from a set of points, we found a modification of the greedy triangulation to be the most suitable for our purposes. Both the Delaunay and greedy triangulations are approximations to the minimum weight triangulation of the plane [Preparata and Shamos 85]. Because they seek to solve a planar problem, they have deficiencies when used in TIN generation, a 3 dimensional problem. One such deficiency is the phenomenon of contour crossing.

Contour crossing occurs when a triangulation uses a short edge that cuts through a ridge or bridges a valley instead of a longer edge that roughly follows the ridge line. In the example shown in figure 1, suppose that the triangulator chooses the solid lines which form a diamond around the ridge. The triangulator must then select between edges A and B. Edge B, which runs across the ridge line, is obviously a better approximation of the terrain, but a triangulator which seeks to minimize total edge length will always choose edge A. To avoid this, we modify the weight assigned to candidate edges. Instead of simply using the length of each edge, we add a measure of how well that edge approximates the terrain. This measure is obtained by summing the square of the vertical difference between the candidate edge and the real terrain at each point along the edge. We can then use a greedy triangulation algorithm to try to minimize this modified weight. The greedy triangulation proceeds as before, sorting the edges by their altered weights and adding edges to the triangulation in increasing order of weight if they do not intersect any edge already added. This altered weight is not a distance, because it does not, in general, obey the triangle rule. This defect makes the Delaunay triangulation inappropriate in this case.

2.4. Termination

The iteration ends when we no longer have any contours from which to generate points, i.e., there are no longer any contours which are at least *minarea* pixels in area. The termination is area-based since we only wish to correct regions which have sufficiently large coverage of the terrain. Additionally, we terminate if δ falls below ε, i.e., the maximum error in our TIN is lower than the error threshold.

3. Experimental Results

In this section we present experimental results for our iterative TIN generation system for two test areas. The terrain shown in Figures 2 and 14 are each composed of a 15 minute quadrangle in Yellowstone National Park. The data is from the Defense Mapping Agency's (DMA) DTED database (level I) which consists of one elevation post every 3 arc seconds. This corresponds to approximately a 100 foot (30m) spacing. The DEM's are shown by simply displaying each one as an intensity image. The image's dynamic range has been linearly reduced to 8 bits for ease of manipulation and display. Bright regions represent areas of high elevation; dark regions represent areas of low elevation. The top of each image is oriented to the northern edge of the DEM.

For each of the example TIN's, the original source data is reduced from a rectangular grid of 90,000 points in a DEM to approximately a 700 point TIN. The actual reduction in data storage size is not quite as dramatic, since each point no longer encodes a single value (elevation); each point now stores x, y, and z coordinates as well as connectivity information. Using a simple three-dimensional point encoding scheme which was not optimized for TIN storage, our first TIN used only 19K, compared to the original 101K used by the DEM. Other experiments have indicated better reduction factors, in the same size areas with less topographic variation we have generated TIN's with approximately 100 points. Obviously, in areas that are quite flat, the algorithm will not generate more points than necessary to maintain the specified error budget.

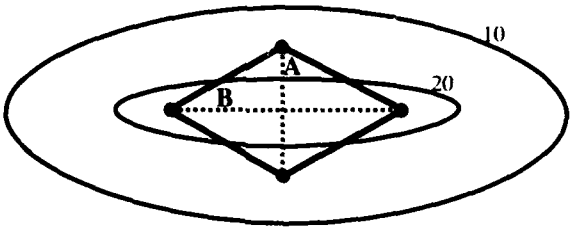


Figure 1: Contour Crossing

3.1. Yellowstone TED11144_7

For this example, the desired minimum horizontal (x and y) distance between points (*mindist*) was set to 50. The minimum vertical (z) distance between points (*minheight*) was set to 5 elevation units. The minimum allowable size for an error polygon (*minarea*) was set to 50 pixels, and the desired maximum error at a point (ε) is 10 elevation units.

The TIN is initialized to the corner points of the DEM. We then interpolate and find the absolute difference between this and the original image. The result is shown in Figure 3. Here the lighter areas are those of higher error. We contour this difference image, find maximal contours, perform a medial axis transform, select points on the approximated medial axis which represent the error polygons, and triangulate again, producing Figure 4.

Even with the first correction, some of the structure begins to appear. The first iteration captured the rough relative heights of each quadrant as well as the fact that there is a valley running down the middle of the terrain. A difference image shown in Figure 5 is generated from this first iteration, in which the areas missed in this iteration are readily identified. For example, by examining the lower right corner of the difference image, we can see that the ridge was picked up well, but we notice that the valley enclosed by the ridge needs more points, which we obtain on the next iteration as shown in Figure 6. With the third and succeeding corrections, the picture becomes increasingly sophisticated. Iteration 3, shown in Figure 7, further develops the central river valley, and begins to draw out another river valley in the southwest corner. This valley is refined on the next iteration (Figure 8). The algorithm then slows down gradually because the points chosen to represent error regions are too close to points already in the TIN. This reaches an extreme during iteration 7, where only one point is added. In the next iteration, no candidate points satisfy the adjacency criterion, and so all 155 of them are added. The algorithm continues in a similar manner, eventually representing all maximal contours on iteration 12 as shown in Figure 12.

From Figure 13, the final difference image, it can be seen that all of the large structures have been captured. Indeed, looking at it, one is hard pressed to find any structure at all. What remains are primarily pieces of river valleys that were too small to be considered important, i.e., error regions below the *minarea* size cutoff.

In terms of absolute error, the worst defects are to be found along the edges. Since we start with the corners in the image, boundaries are initially represented by a smooth slope connecting the corners. This generally poor approximation persists until a point is added at the boundary, but this does not generally happen, because the medial axis tends to be in the center of the error regions. Thus the area adjacent to the boundary is represented by a few long, thin triangles which poorly approximate the terrain.

3.2. Yellowstone TED11043_6

A second test area having sharp ridges and deep valleys provides an especially challenging test for the triangulator. It is difficult to keep the triangulator from making contour crossing errors, which would chop holes in ridges or fill in valleys. The modifications to the greedy triangulation algorithm discussed in Section 2.3 reduce the magnitude of the problem, but occasionally the triangulator has no choice but to cross contour lines. This example shows how well the iterative algorithm works in such an area.

We use the same values for the control parameters as in our previous example. The algorithm begins with the four corner points, and quickly picks up major terrain features. In Figure 15, the algorithm begins to resolve some of the finer features, and here the triangulator is forced to make a number of contour crossing errors. Additional points are added to correct for this in Figure 16, and the approximation becomes more recognizable as the original DEM terrain image. As the run continues, finer and finer details from the original are represented in the approximation. By Figure 18 the approximation is very close to the original, but several more iterations are required to meet the error bound and refine more subtle terrain variations.



Figure 2: Original DEM, 300 x 300 Points

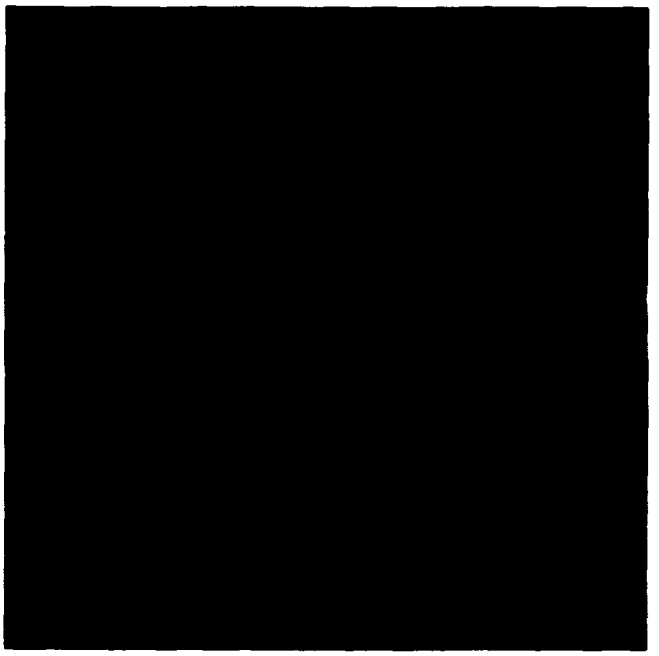


Figure 3: Initial Difference DEM



Figure 4: Iteration 1 DEM, 35 vertices

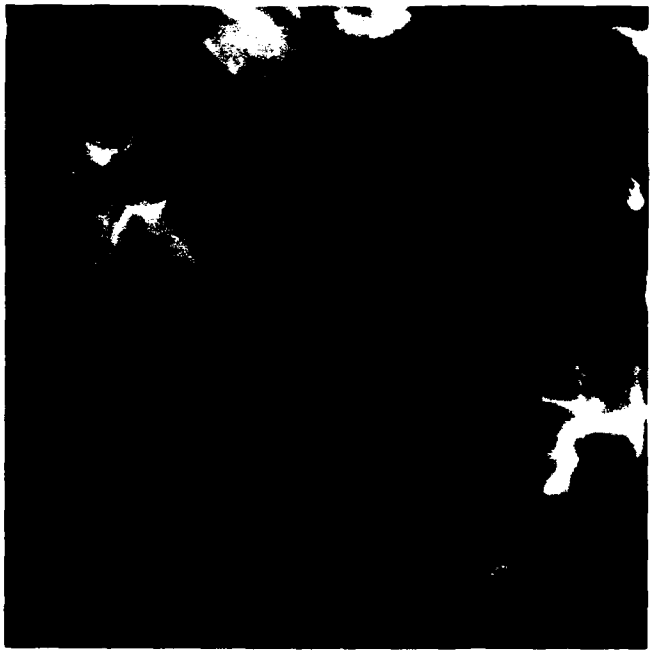


Figure 5: Difference DEM 2

This area, TED11043_6, is about as complex as TED11144_7 in terms of the number of points the algorithm needs to approximate it within the error budget. It is apparent, however, that it differs widely from TED11144_7 in terms of overall appearance. Whereas TED11144_7 had rounded and smooth ridges, this example has sharp peaks and deep valleys. Each required additional points for different reasons corresponding to the differences in terrain. TED11144_7 needed points because the TIN approximation produced a faceted appearance, so additional facets were needed to represent curves. In this experiment, facets were ideally suited to the jagged nature of the terrain, but extra points were needed as hints to the triangulator because of the steepness of the terrain.

3.3. Numerical Results

In this section we consider the performance of the algorithm numerically for each of the test cases. Recall that the images were scaled to an 8-bit range, so the statistics presented here are given in terms of pixels. Tables 1 and 2 represent performance statistics

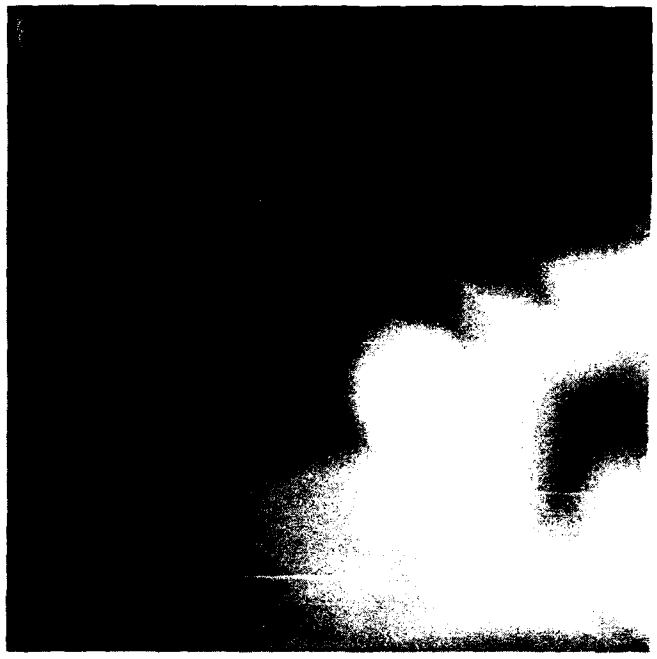


Figure 6: Iteration 2 DEM, 159 vertices



Figure 7: Iteration 3 DEM, 286 vertices



Figure 8: Iteration 4 DEM, 354 vertices



Figure 9: Iteration 6 DEM, 381 vertices

for TED11144_7 and TED11043_6, respectively. In each Table, the number of points used in each iteration is summarized and the RMS error is given. The internal RMS error is an attempt to remove the edge effect from error measurement. This column gives the RMS error excluding a 5% (15 pixel) border around the edges.

Maximum error (δ) gives the highest numbered point on the error histogram which has more than a given number (*minarea*, which is in this case 50) of pixels. Maximal contours gives the number of contours above ε that are more than *minarea* pixels in area. The algorithm will terminate either if the maximum error drops below ε or if there are no maximal contours of the requisite size left. In both cases, the algorithm terminated because there were no more maximal contours. This happens with increasing frequency as the value of ε relative to the dynamic range of the terrain decreases. The edge effect is quite visible in the last few iterations of the algorithm, as evidenced by the increase in the difference between the RMS error and



Figure 10: Iteration 8 DEM, 537 vertices

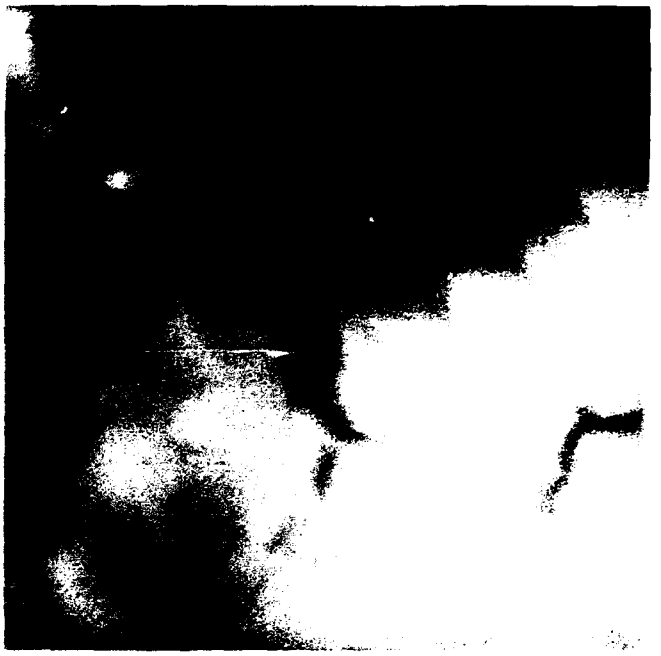


Figure 11: Iteration 10 DEM, 643 vertices

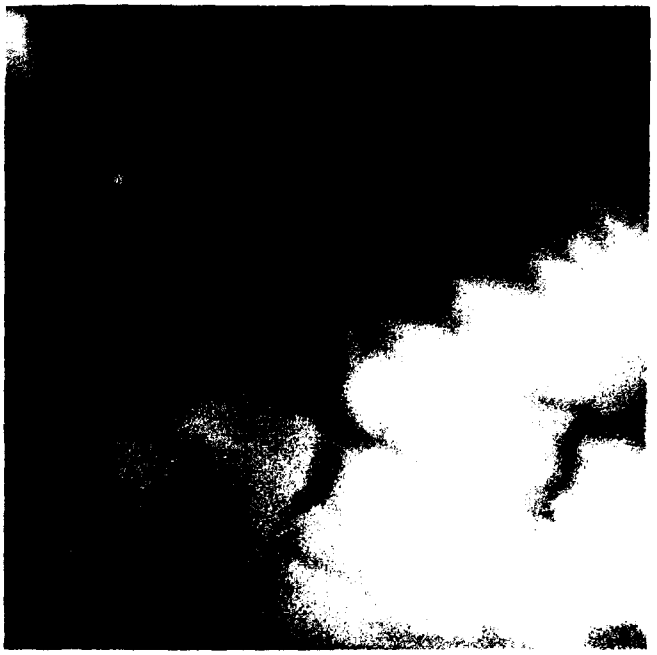


Figure 12: Iteration 12 DEM, 686 vertices

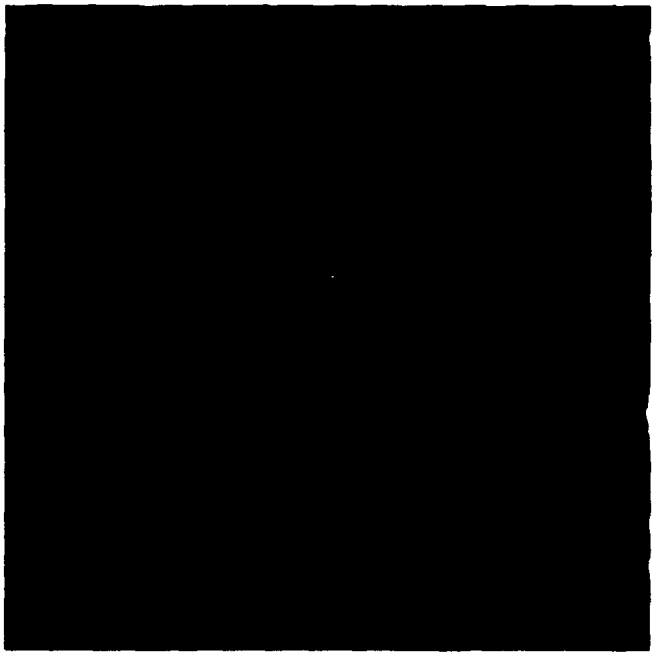


Figure 13: Final Difference DEM

the internal RMS error, the latter of which does not include a 15-pixel border around the edges.

3.4. Triangulation

Another way to visualize the results of the TIN network is to display the triangulations of the selected points. Figures 22 and 23 are the networks generated for TED11144_7 during the second and twelfth iterations. They illustrate the clustering of points in areas where the underlying terrain exhibits high variability in elevation, as is expected. The edge effect is visible, mainly along the lower and right edges. It also shows, to some degree, the success that the triangulator has in generating edges which follow the terrain. Such terrain following causes the narrow triangles which can be seen along the rivers.

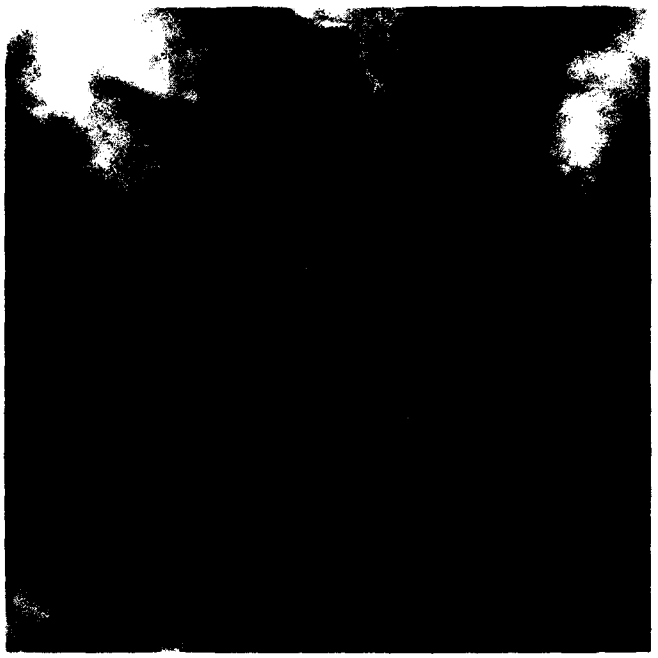


Figure 14: Original DEM, 300 x 300 Points

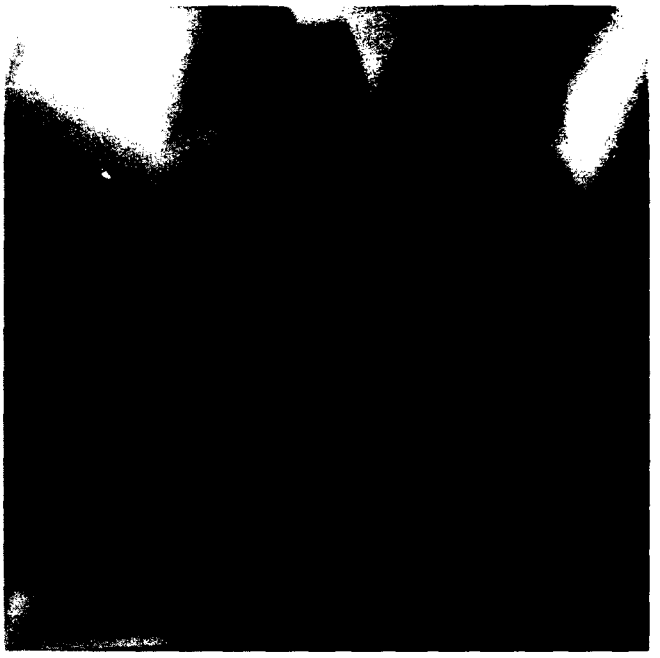


Figure 15: Iteration 2 DEM, 172 vertices

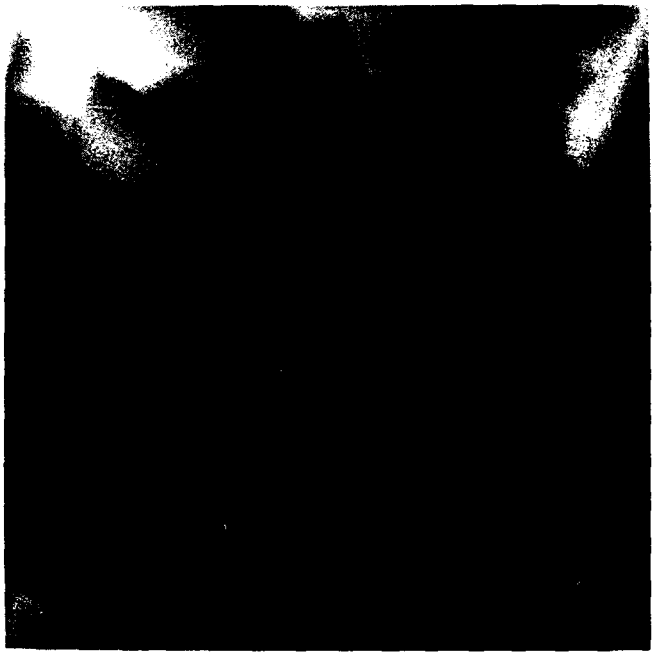


Figure 16: Iteration 3 DEM, 309 vertices

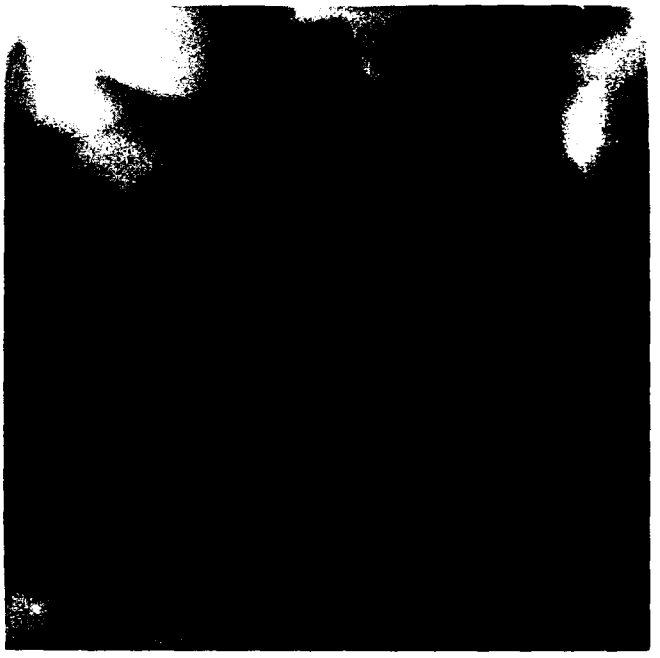


Figure 17: Iteration 4 DEM, 388 vertices

4. Future Work

Iterative TIN construction is an area that is rich in possibilities for future work. Interesting research issues include how to integrate specific topographic features as constraints; the tiling of individual TIN's to provide a larger area of coverage; and the use of cartographic feature data to improve the correlation between terrain and object data.

In our technique, we select points solely based upon our contour error model. It is clear that analysis of the topography can yield points such as peaks, valleys, and ridges that can be used to augment the current selection process. How to select a subset of these points, without greatly increasing the number of points in the TIN, and while continuing to minimize the overall error, raises an interesting set of selection and optimization problems.

The TIN's generated in this paper cannot be connected simply by placing them side-by-side because of the edge effects. Simply, there are no guarantees that

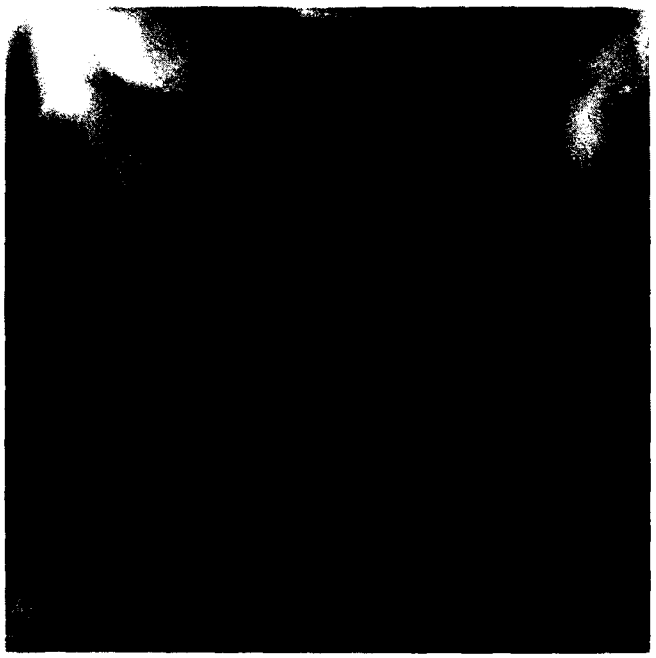


Figure 18: Iteration 5 DEM, 418 vertices

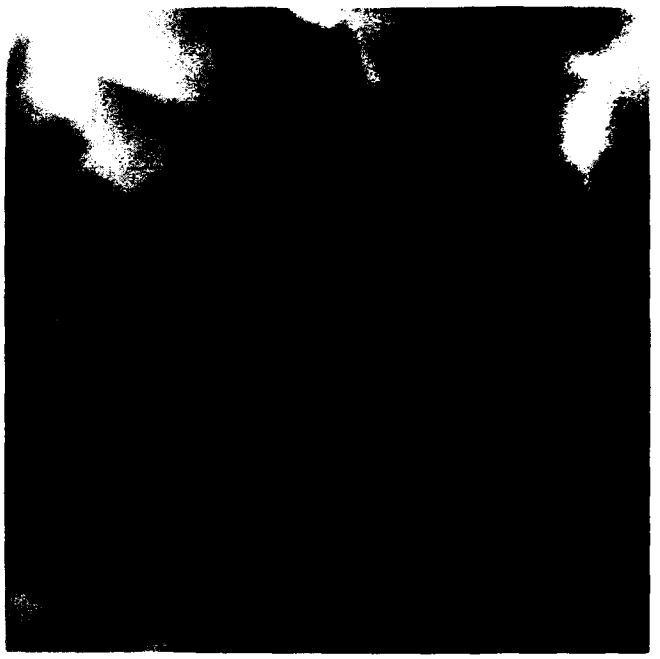


Figure 19: Iteration 7 DEM, 597 vertices

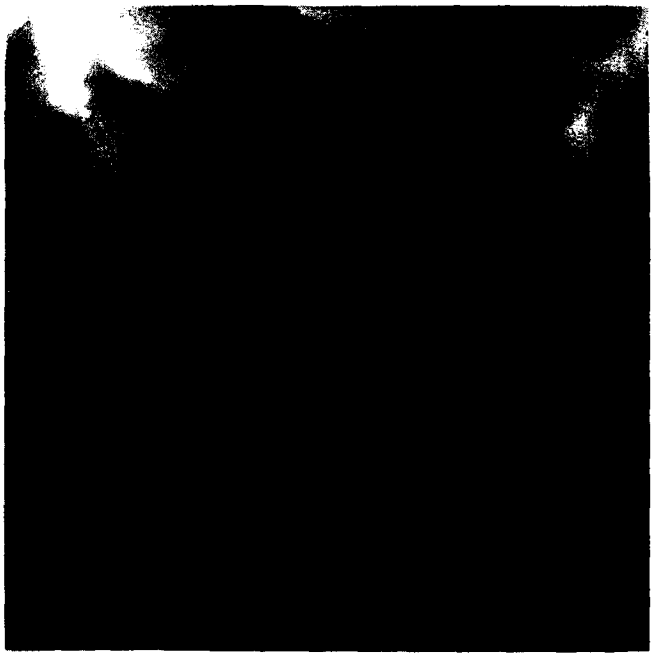


Figure 20: Iteration 9 DEM, 689 vertices

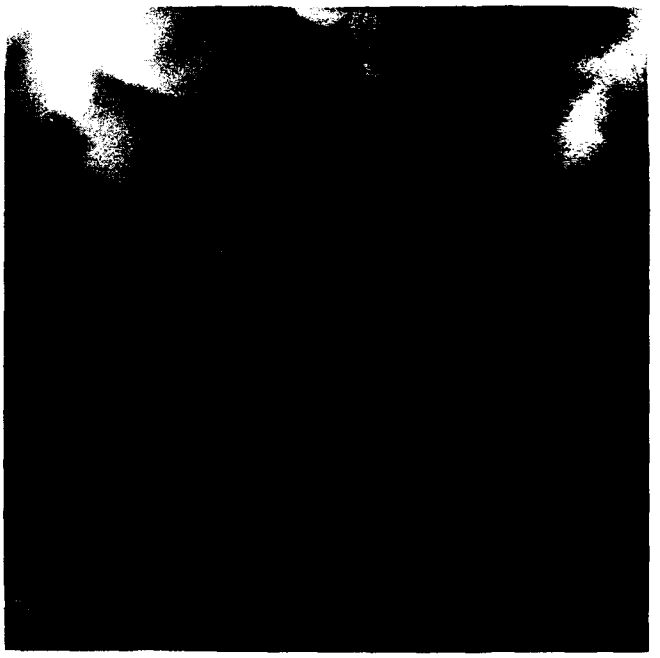


Figure 21: Iteration 10 DEM, 713 vertices

adjacent terrain patches will not have severe discontinuities at the borders. Instead, some way of meshing TIN's generated independently, such as re-triangulating along the edges, is needed. The generation of hierarchical TIN's using an iterative method are another interesting area for research. Such representations allow the resolution of the TIN to vary by nesting an additional higher resolution TIN within a TIN at the previous level of the hierarchy. Both of these issues arise due to concerns to support terrain models for large areas of coverage and with variable levels-of-detail.

Finally, the use of digital spatial databases containing road networks, drainage networks, and specific man-made objects in conjunction with DEM's or TIN's raise the issue of correlation between the spatial location of features and the underlying terrain. A constraint that could be applied during point selection would be to maintain coherency between the location of spatial database objects and the TIN. For example, one would expect that streams would continue to flow

Performance Results for TED11144_7					
Iteration	TIN Points	Global RMS Error	Internal RMS Error	Maximum Error	Maximal Contours
0	4	42.9111	42.9195	113	8
1	35	31.8709	31.5403	84	28
2	159	14.9189	15.0460	44	59
3	286	8.11141	7.65872	29	74
4	354	7.04317	6.59736	28	64
5	375	6.65025	6.36294	27	59
6	381	6.54591	6.26490	27	57
7	382	6.54332	6.26490	27	56
8	537	5.37215	5.00352	22	41
9	547	5.25415	4.86895	22	35
10	643	4.76598	4.41236	20	17
11	646	4.68777	4.31051	20	14
12	686	4.47456	4.11826	19	0

Table 1: Numerical Accuracy TED11144_7

Performance Results for TED11043_6					
Iteration	TIN Points	Global RMS Error	Internal RMS Error	Maximum Error	Maximal Contours
0	4	66.0866	67.2875	132	15
1	65	23.1250	21.0115	75	30
2	172	12.2026	11.8442	43	57
3	309	8.14550	7.65331	33	73
4	388	7.17465	6.60838	28	69
5	418	6.62902	6.15420	27	61
6	428	6.50469	5.99604	27	54
7	597	5.40780	5.14117	21	32
8	600	5.39088	5.11934	20	30
9	689	4.99839	4.72627	18	8
10	713	4.87780	4.57424	18	0

Table 2: Numerical Accuracy TED11043_6

5. Conclusions

We described a new technique to produce a triangulated irregular network (TIN) from a digital elevation model (DEM). Our overall goal is to produce an approximate terrain description that preserves the major topographic features using a

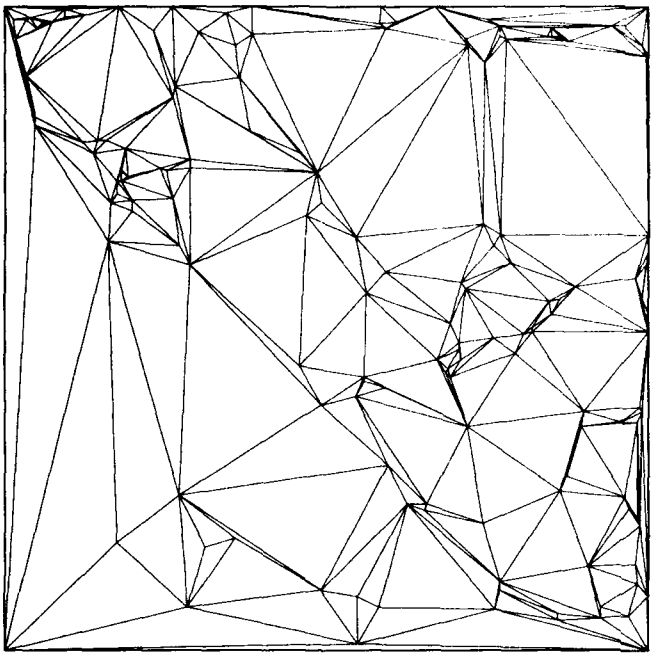


Figure 22: Iteration 2 Triangulation

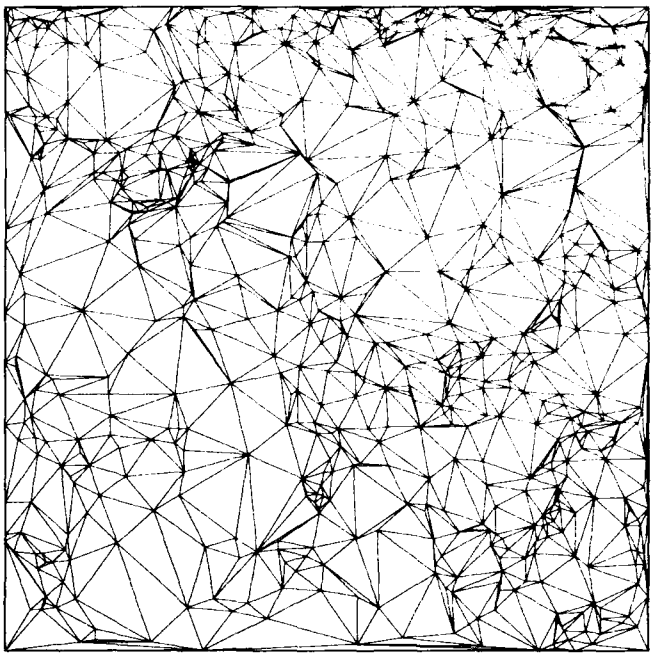


Figure 23: Final Triangulation

important for computer image generation, but also emerge in simulation and autonomous navigation using digital maps.

5. Conclusions

We described a new technique to produce a triangulated irregular network (TIN) from a digital elevation model (DEM). Our overall goal is to produce an approximate terrain description that preserves the major topographic features using a greatly reduced set of points selected from the original DEM. The TIN generation process is iterative; at each iteration we identify areas in the DEM that lie outside of a user-supplied error tolerance in the TIN, and attempt to choose points in the DEM to more accurately model these areas. Point selection involves the computation of the difference between the actual DEM and an approximate DEM. This approximate DEM is calculated by interpolating elevation points from the TIN.

The iterative nature of the algorithm permits users to terminate the terrain approximation algorithm based on operational criteria, such as median error, maximal error, or number of points used to construct the TIN. This is particularly relevant to real-time computer image generation as various scene rendering systems utilizing polygonal terrain patches have well defined limitations in order to maintain real-time performance. It also implies that the TIN generation procedure is automatically sensitive to smooth or rough terrain in that it selects only enough points to satisfy the required error constraints and tends to place points in those areas having the greatest topographic complexity.

Acknowledgements

Yuan Hsieh collaborated on an earlier version of this TIN generation system. This paper benefited from careful reading by Jefferey Shufelt. We thank the members of the Digital Mapping Laboratory for not shooting at the first author while he was debugging some of the triangulation software.

References

[Aviad 88]
Aviad, Z.
Locating Corners in Noisy Curves by Delineating Imperfect Sequences.
Technical Report CMU-CS-88-199, Carnegie-Mellon University, December, 1988.

[Carter 88]
James R. Carter.
Digital Representations of Topographic Surfaces.
Photogrammetric Engineering and Remote Sensing 54(11):1577-1580, November, 1988.

[Chen and Guevara 87]
Zi-Tan Chen and J. Armando Guevara.
Systematic Selection of Very Important Points (VIP) from Digital Terrain Model for Constructing Triangular Irregular Networks.
In Nicholas R. Chrisman (editor), *Proceedings of the Eighth International Symposium on Computer-Assisted Cartography*, pages 50-54. 1987.

[Mark 75]
David M. Mark.
Computer analysis of topography: a comparison of terrain storage methods.
Geographiska Annaler 57A:179-188, 1975.

[Peucker, et. al. 76]
Thomas K. Peucker, Robert J. Fowler, James J. Little, and David M. Mark.
Digital Representation of Three Dimensional Surfaces By Triangulated Irregular Networks (TIN).
Technical Report 10, Office of Naval Research, Geography Programs, 1976.

[Peucker, et. al. 78]
T.K. Peucker, R.J. Fowler, James J. Little, and David M. Mark.
The Triangulated Irregular Network.
In *Proceedings of the ASP Digital Terrain Model Symposium.* 1978.

[Preparata and Shamos 85]
F. P. Preparata and M. I. Shamos.
Computational Geometry.
Springer-Verlag, 1985.

A Novel Image Registration Algorithm*

Qinfen Zheng
Signal and Image Processing Institute
3740 McClintock Avenue
University of Southern California
Los Angeles, CA 90089-2564

Rama Chellappa
Department of Electrical Engineering
Center for Automation Research and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742-3411

Abstract

A computational vision approach is presented for the estimation of 2-D translation, rotation, and scale from two partially overlapping images. The approach results in a fast and novel method that produces excellent results even when large rotation and scale changes have occurred between the two frames and the images are devoid of significant features. In our method an illuminant direction estimation method is first used to obtain an initial estimate of camera rotation. A small number of feature points are then located based on a Gabor wavelet model for detecting local curvature discontinuities. An initial estimate of scale and translation is obtained by pairwise matching of the feature points detected from both frames. Finally, hierarchical feature matching is performed to obtain an accurate estimate of translation, rotation and scale. Experiments with synthetic and real images show that this algorithm yields accurate results when the scale between the image pair differ by up to 10%, the overlap between the two frames is as small as 35%, and the camera rotation between the two frames is significant. Applications of the method to texture and stereo image registration, satellite image mosaicking, and moving object detection are presented.

1 Introduction

Automatic image registration is an important problem in computer vision and image processing. Traditional solutions [1, 5, 10, 12, 13, 18] to this problem are unreliable when the rotation of the camera and scale change between the two frames are significant. Registration becomes even more difficult if the images are devoid of significant features and/or the overlap between the two frames is small. In this paper we present a computational vision approach for the estimation of 2-D translation, rotation, and scale from two partially overlapped images. Figure 1 shows the block diagram of our camera motion estimation algorithm. We notice that the illuminant direction is inherent to environment and is invariant to camera motion. The difference in illuminant direction of two successive frames is equal to the amount of camera rotation. By estimating the illuminant direction in each frame, we estimate the rotation between the two frames and simplify the matching process. To estimate the illuminant azimuth, we use a local voting estimator reported in [20]. Since the common area between the two frames can be much smaller than the image field and additionally there is scaling between the two frames, methods based on correlation matching become unreliable. In this work we use a feature based matching technique. First we extract a small number of feature points based on a Gabor wavelet model for local curvature analysis. In doing this we compute a local energy measure defined as the interaction of image convolved with basic wavelet functions at different scales, and then detect the locations of local maxima of such an energy map as the feature points. The effect of local inhibition of nearby feature points is also considered. Since no prior knowledge about the translation is available, an initial estimate of scale and translation is obtained by pairwise matching between the neighbors of feature points detected from both the frames. Subsequently, hierarchical correlation matching is performed to obtain an accurate camera motion estimate. Using this algorithm we have obtained impressive results for several applications including stereo and texture image registration, satellite image mosaicking, and moving object detection.

The organization of the paper is as follows: Section 2 discusses the basic steps used in our matching algorithm, issues of camera rotation estimation, feature extraction, matching criterion, scale and translation estimation, and hierarchical implementation are addressed. Section 3 presents the matching algorithm. Section 4 presents experimental results on texture image registration, stereo matching, satellite image mosaicking, and moving object detection. The work is summarized in Section 5.

2 Basic Steps

Before a discussion of our algorithm, some definitions and basic steps used should be addressed.

*The support of the Defense Advanced Research Projects Agency (ARPA Order No. 6989) and the U.S. Army Engineer Topographic Laboratories under Contract DACA 76-89-C-0019 is gratefully acknowledged.

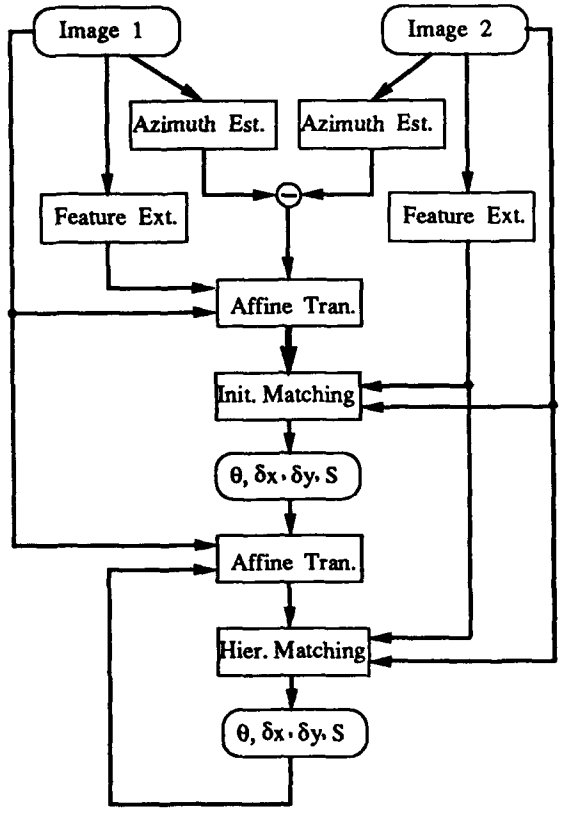


Figure 1: Block diagram of image registration algorithm

Initial Estimation of Camera Rotation

As shown in Figure 1, the initial estimation of camera rotation is computed as the difference of illuminant azimuth angles estimated from both of the frames. To estimate the illuminant direction from the image we use the local voting illuminant azimuth estimator discussed in [20]. To be more specific, for each pixel we first compute

$$\vec{X} = \begin{pmatrix} \tilde{x}_L \\ \tilde{y}_L \end{pmatrix}$$

by solving

$$\vec{X} = (B^t B)^{-1} B^t d\vec{I}$$

where

$$d\vec{I} = \begin{pmatrix} \delta I_1 \\ \delta I_2 \\ \vdots \\ \delta I_N \end{pmatrix},$$

$$B = \begin{pmatrix} \delta x_1 & \delta y_1 \\ \delta x_2 & \delta y_2 \\ \vdots & \vdots \\ \delta x_N & \delta y_N \end{pmatrix},$$

and N is the number of measured directions for $\vec{s}$.

The estimate of illuminant azimuth is then computed as

$$\tau = \arctan \left(\frac{\mathbf{E}_{x,y} \left\{ \frac{\tilde{y}_L}{\sqrt{\tilde{x}_L^2 + \tilde{y}_L^2}} \right\}}{\mathbf{E}_{x,y} \left\{ \frac{\tilde{x}_L}{\sqrt{\tilde{x}_L^2 + \tilde{y}_L^2}} \right\}} \right).$$

After the illuminant azimuth τ_i, $i = 1, 2$ are computed, the initial estimation of camera rotation is computed as

$$\theta_0 = \tau_1 - \tau_2 \tag{1}$$

Feature Detection Feature detection is an important step in our algorithm. In this work, we use a method which is based on a biologically motivated model for identifying local curvature discontinuities, and makes use of Gabor wavelet decomposition of the image and local scale interactions between features [15]. The basic wavelet function used in our decomposition is of the form

$$\Phi_\lambda(x, y, \theta) = e^{-(\lambda^2 x'^2 + y'^2) + i\pi x'} \tag{2}$$

$$x' = x\cos\theta + y\sin\theta$$

$$y' = -x\sin\theta + y\cos\theta$$

where θ is the preferred spatial orientation and λ is the aspect ratio of the Gaussian. For convenience, we will drop the subscripts in further discussion. In all our experiments, λ is set to 1, and θ is discretized into four orientations.

The corresponding wavelet transformation is obtained by convolving the image data $\mathbf{f}$ with a bank of filters whose responses are simple dilations and translations of the basic wavelet in (2), and are denoted by

$$W_j(x, y, \theta) = \mathbf{f} \bigotimes \Phi(\alpha^j x, \alpha^j y, \theta), \; j = \{0, 1, 2, ..\} \tag{3}$$

Here α denotes the scale parameter. Usually the parameter values used are those corresponding to half-octave ($\alpha = \sqrt{2}$) or octave ($\alpha = 2$). Physically, this transformation detects features in the image such as line and step edges. Biologically, this models the processing by simple cells in the visual cortex of mammals. These features by themselves are not good for applications such as obtaining correspondence. The next stage in our feature detection module involves interactions between these simple features (at different spatial frequencies, within each orientation). This step can be identified with the responses of hypercomplex cells in the visual cortex. Hypercomplex cells exhibit end-inhibition. They are sensitive to oriented lines and step edges of short lengths, and their response decreases if the lengths are increased. Using scale interactions to model these cells was first suggested by Hubel and Wiesel [11], who were the first to discover these cells in the visual cortex. Subsequent anatomical studies have also supported this model [2]. We model these interactions as follows:

$$I_{m,n}(x, y) = \max_\theta g(||W_m(x, y, \theta) - \gamma W_n(x, y, \theta)||) \tag{4}$$

where g is a non-linear transformation, such as thresholding or a sigmoid non-linearity, γ is a normalizing factor, and $n > m$. In order to identify features at different scales, one has to consider different scale interactions. The final step is to actually localize these features, and

this is done by looking at the local maxima of these feature responses.

Image Transformation The image transform is implemented by bilinear interpolation from the original image. A 2-D affine transform with parameters (s, θ, ΔX, ΔY) is defined as

$$\begin{pmatrix} X' \\ Y' \end{pmatrix} = s \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} + \begin{pmatrix} \Delta X \\ \Delta Y \end{pmatrix}. \quad (5)$$

For a grid point (X', Y') on the transformed frame its location on the original frame is given by

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \frac{1}{s} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} X' - \Delta X \\ Y' - \Delta Y \end{pmatrix} \quad (6)$$

The four nearest grid points of (X, Y) are $(\underline{X}, \underline{Y})$, $(\overline{X}, \underline{Y})$, $(\underline{X}, \overline{Y})$, and $(\overline{X}, \overline{Y})$, with

$$\begin{aligned} \underline{X} &= \text{int}(X) \\ \underline{Y} &= \text{int}(Y) \\ \overline{X} &= \underline{X} + 1 \\ \overline{Y} &= \underline{Y} + 1 \end{aligned}$$

where int$(\cdot)$ is the floor function defined as the largest integer not greater than the variable. The image value at (X', Y') is computed by a bilinear interpolation[1] as

$$\begin{aligned} \mathbf{f}'(X', Y') &= \mathbf{f}(X, Y) \\ &= \overline{\varepsilon}_x \overline{\varepsilon}_y \mathbf{f}(\underline{X}, \underline{Y}) + \underline{\varepsilon}_x \overline{\varepsilon}_y \mathbf{f}(\overline{X}, \underline{Y}) \\ &\quad + \overline{\varepsilon}_x \underline{\varepsilon}_y \mathbf{f}(\underline{X}, \overline{Y}) + \underline{\varepsilon}_x \underline{\varepsilon}_y \mathbf{f}(\overline{X}, \overline{Y}) \quad (7) \end{aligned}$$

where

$$\begin{aligned} \underline{\varepsilon}_x &= X - \underline{X} \\ \overline{\varepsilon}_x &= \overline{X} - X \\ \underline{\varepsilon}_y &= Y - \underline{Y} \\ \overline{\varepsilon}_y &= \overline{Y} - Y \end{aligned}$$

Matching Criterion The goodness of fit between pixels $\mathbf{f}_1(m, n)$ in frame t_1 and $\mathbf{f}_2(u, v)$ in frame t_2 is measured by their mutual correlation coefficient

$$\psi_{f_1 f_2}(m, n; u, v) = \frac{1}{\sigma_1 \sigma_2 (2\omega_m + 1)^2} \times \sum_{i,j=-\omega_m}^{i,j=\omega_m} (\mathbf{f}_1(m+i, n+j) - \mu_1)(\mathbf{f}_2(u+i, v+j) - \mu_2) \quad (8)$$

where

$$\mu_1 = \frac{1}{(2\omega_m + 1)^2} \sum_{i,j=-\omega_m}^{i,j=\omega_m} \mathbf{f}_1(m+i, n+j)$$

$$\mu_2 = \frac{1}{(2\omega_m + 1)^2} \sum_{i,j=-\omega_m}^{i,j=\omega_m} \mathbf{f}_2(u+i, v+j)$$

$$\sigma_1 = \sqrt{\frac{1}{(2\omega_m + 1)^2 - 1} \sum_{i,j=-\omega_m}^{i,j=\omega_m} \mathbf{f}_1(m+i, n+j)^2 - \mu_1^2}$$

$$\sigma_2 = \sqrt{\frac{1}{(2\omega_m + 1)^2 - 1} \sum_{i,j=-\omega_m}^{i,j=\omega_m} \mathbf{f}_2(u+i, v+j)^2 - \mu_2^2}$$

[1] Here we assume that (X, Y) is an inner point. For matching purposes, boundary and out of the image frame pixels can simply be ignored.

and $(2\omega_m + 1)^2$ is the area of the matching window.

Initial Matching Initial matching is determined by best pairwise fitting between feature points detected from different frames. Assume that the feature points detected from frame t_1 are $\{f_1(X_{1i}, Y_{1i}), i = 1, \ldots, N_1\}$ and the feature points detected from frame t_2 are $\{f_2(\hat{X}_{2i}, \hat{Y}_{2i}), i = 1, \ldots, N_2\}$, then the feature point $f_1(X_{1i}, Y_{1i})$ is matched to $f_2(\hat{X}_{1i}, \hat{Y}_{1i})$ of frame t_2 if

$$\psi_{f_1 f_2}(X_{1i}, Y_{1i}; \hat{X}_{1i}, \hat{Y}_{1i}) = \max_{1 \le j \le N_2} \max_{\substack{|u - X_{2j}| \le \omega_s \\ |v - Y_{2j}| \le \omega_s}} \psi_{f_1 f_2}(X_{1i}, Y_{1i}; u, v)$$

where ω_s is the search window parameter with $(2\omega_s + 1)^2$ be the size of searching area. Similarly, the feature point $f_2(\hat{X}_{2i}, \hat{Y}_{2i})$ is matched to $f_1(X_{2i}, Y_{2i})$ of frame t_1 if

$$\psi_{f_1 f_2}(X_{2i}, Y_{2i}; \hat{X}_{2i}, \hat{Y}_{2i}) = \max_{1 \le j \le N_1} \max_{\substack{|m - X_{1j}| \le \omega_s \\ |n - Y_{1j}| \le \omega_s}} \psi_{f_1 f_2}(m, n; \hat{X}_{2i}, \hat{Y}_{2i}).$$

Scale Estimation Since the Euclidean distance between the feature points only depends on the scale between the two frames, and is invariant to rotation and translation, the scale factor can be estimated prior to the estimation of other parameters. Assume that the matched feature point pairs are $\{(X_i', Y_i') \Rightarrow (\hat{X}_i, \hat{Y}_i)\ i = 1, \ldots, N'\}$ with N' be the number of matched feature pairs. Then

$$(\hat{d}_1 \quad \hat{d}_2 \quad \cdots \quad \hat{d}_{N'}) = s' (d_1 \quad d_2 \quad \cdots \quad d_{N'}) \quad (9)$$

where

$$\begin{aligned} d_i &= \sqrt{(X_i' - \overline{X_i'})^2 + (Y_i' - \overline{Y_i'})^2} \\ \hat{d}_i &= \sqrt{(\hat{X}_i - \overline{\hat{X}_i})^2 + (\hat{Y}_i - \overline{\hat{Y}_i})^2} \end{aligned}$$

$$\begin{aligned} \overline{X_i'} &= \frac{1}{N'} \sum_{i=1}^{N'} X_i' \\ \overline{Y_i'} &= \frac{1}{N'} \sum_{i=1}^{N'} Y_i' \\ \overline{\hat{X}_i} &= \frac{1}{N'} \sum_{i=1}^{N'} \hat{X}_i \\ \overline{\hat{Y}_i} &= \frac{1}{N'} \sum_{i=1}^{N'} \hat{Y}_i \end{aligned}$$

The scale factor from frame t_1 to t_2 is computed as

$$s' = \frac{\sum_{i=1}^{N'} d_i \cdot \hat{d}_i}{\sum_{i=1}^{N'} d_i \cdot d_i}. \quad (10)$$

Rotation and Translation Estimation With the scale factor determined, the rotation and translation between the transformed frame t_1 and frame t_2 can

be computed as follows. Assuming that the matched feature point pairs are $\{(X'_i, Y'_i) \Rightarrow (\hat{X}_i, \hat{Y}_i)\ i = 1, \ldots, N'\}$, the relation between the matched feature pairs is

$$\begin{pmatrix} \hat{X}_i \\ \hat{Y}_i \end{pmatrix} = s' \begin{pmatrix} \cos\theta' & \sin\theta' \\ -\sin\theta' & \cos\theta' \end{pmatrix} \begin{pmatrix} X'_i \\ Y'_i \end{pmatrix} + \begin{pmatrix} \Delta X' \\ \Delta Y' \end{pmatrix}, \quad (11)$$

Note θ', being the residual of initial rotation estimation, is very small. By approximating $\cos\theta'$ and $\sin\theta'$ up to linear terms, (11) can be rewritten as

$$\begin{pmatrix} \hat{X}_i \\ \hat{Y}_i \end{pmatrix} = \begin{pmatrix} 1 & \theta' \\ -\theta' & 1 \end{pmatrix} \begin{pmatrix} s'X'_i \\ s'Y'_i \end{pmatrix} + \begin{pmatrix} \Delta X' \\ \Delta Y' \end{pmatrix}, \quad (12)$$

or

$$\vec{A} = B\vec{C} \quad (13)$$

where

$$\vec{A} = \begin{pmatrix} \hat{X}_1 - s'X'_1 \\ \hat{Y}_1 - s'Y'_1 \\ \vdots \\ \hat{X}_{N'} - s'X'_{N'} \\ \hat{Y}_{N'} - s'Y'_{N'} \end{pmatrix};$$

$$B = \begin{pmatrix} s'Y'_1 & 1 & 0 \\ -s'X'_1 & 0 & 1 \\ \vdots & \vdots & \vdots \\ s'Y'_{N'} & 1 & 0 \\ -s'X'_{N'} & 0 & 1 \end{pmatrix};$$

$$\vec{C} = \begin{pmatrix} \theta' \\ \Delta X' \\ \Delta Y' \end{pmatrix}.$$

The vector $\vec{C}$ can then be computed as

$$\vec{C} = (B^t B)^{-1} B^t \vec{A}. \quad (14)$$

Matching Refinement After initial matching is accomplished, the matching is refined on every resolution layer of the matching pyramid. To do this, frame t_1 and its feature points are first transformed using the estimated parameters. The transformed feature points are then truncated to the nearest grid location. Let the transformed feature point be $f'_1(X'_{1i}, Y'_{1i})$; it is matched to $f_2(\hat{X}'_{1i}, \hat{Y}'_{1i})$ on frame t_2 if

$$\psi_{f'_1 f_2}(X'_{1i}, Y'_{1i}; \hat{X}'_{1i}, \hat{Y}'_{1i}) = \max_{\substack{|m| \le \omega_s \\ |n| \le \omega_s}} \psi_{f'_1 f_2}(X'_{1i}, Y'_{1i}; X'_{1i} + m, Y'_{1i} + n) \quad (15)$$

Estimation Refinement In our method, at each level of the matching hierarchy, the feature points from frame t_1 $\{f_1(X_{1i}, Y_{1i}), i = 1, \ldots, N_1\}$ are first transformed using (5) to get $\{f'_1(X'_{1i}, Y'_{1i}), i = 1, \ldots, N'_1\}$. The transformed feature points are then matched to frame t_2 and the correction on initial estimation $(s', \theta', \Delta X', \Delta Y')$ is computed using (10) and (14). After that, the total transformation is obtained by combining the two transformations together:

$$\begin{aligned} \begin{pmatrix} \hat{X} \\ \hat{Y} \end{pmatrix} &= s' \begin{pmatrix} \cos\theta' & \sin\theta' \\ -\sin\theta' & \cos\theta' \end{pmatrix} \begin{pmatrix} X' \\ Y' \end{pmatrix} + \begin{pmatrix} \Delta X' \\ \Delta Y' \end{pmatrix} \\ &= s' \begin{pmatrix} \cos\theta' & \sin\theta' \\ -\sin\theta' & \cos\theta' \end{pmatrix} \left[s \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} + \begin{pmatrix} \Delta X \\ \Delta Y \end{pmatrix} \right] + \begin{pmatrix} \Delta X' \\ \Delta Y' \end{pmatrix} \\ &= s\,s' \begin{pmatrix} \cos(\theta + \theta') & \sin(\theta + \theta') \\ -\sin(\theta + \theta') & \cos(\theta + \theta') \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} \\ &\quad + \begin{pmatrix} s'(\cos\theta'\,\Delta X + \sin\theta'\,\Delta Y) + \Delta X' \\ s'(-\sin\theta'\,\Delta X + \cos\theta'\,\Delta Y) + \Delta Y' \end{pmatrix} \end{aligned} \quad (16)$$

So the estimates are updated using

$$\begin{pmatrix} \theta + \theta' \\ s\,s' \\ s'(\cos\theta'\,\Delta X + \sin\theta'\,\Delta Y) + \Delta X' \\ s'(-\sin\theta'\,\Delta X + \cos\theta'\,\Delta Y) + \Delta Y' \end{pmatrix} \Longrightarrow \begin{pmatrix} \theta \\ s \\ \Delta X \\ \Delta Y \end{pmatrix} \quad (17)$$

3 Matching Algorithm

Combining the operations discussed in Section 2, we obtain our matching algorithm, summarized as

Step 1:
- Estimate the illuminant azimuth τ_i from frame t_i, $i = 1, 2$;
- Set
$$\begin{cases} s &= 1 \\ \theta &= \tau_1 - \tau_2 \\ \Delta X &= 0 \\ \Delta Y &= 0 \end{cases}$$

Step 2:
- Reduce the image size to that of the lowest resolution layer;
- Estimate feature points from each frame.

Step 3:
- Apply an affine transformation with parameters $(s, \theta, \Delta X, \Delta Y)$ on the lowest resolution version of frame t_1 and its feature points;
- Do initial matching to obtain estimates $(s', \theta', \Delta X', \Delta Y')$;
- Update $(s, \theta, \Delta X, \Delta Y)$ by (17).

Step 4:
- Reduce the image resolution corresponding to current layer of the matching hierarchy;
- Magnify the coordinates of the feature points corresponding to the resolution of the current layer;
- Apply an affine transform with parameter $(s, \theta, \Delta X, \Delta Y)$ to frame t_1 and its feature points;
- Do matching refinement to obtain $(s', \theta', \Delta X', \Delta Y')$;
- Update the estimates using (17).

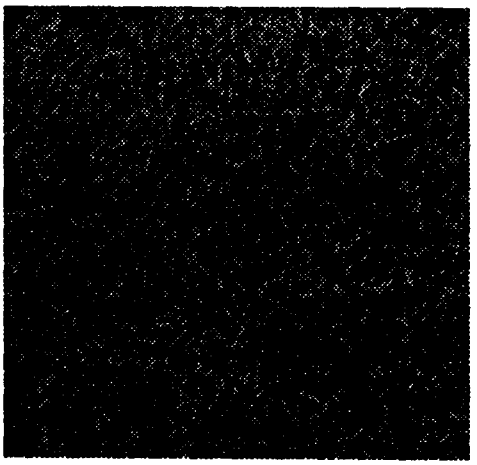
Step 5:
- If the current level is at the highest resolution then stop;
- Increase the image resolution and adjust the translation estimation by

$$\begin{pmatrix} 2\Delta X \\ 2\Delta Y \end{pmatrix} \Longrightarrow \begin{pmatrix} \Delta X \\ \Delta Y \end{pmatrix};$$

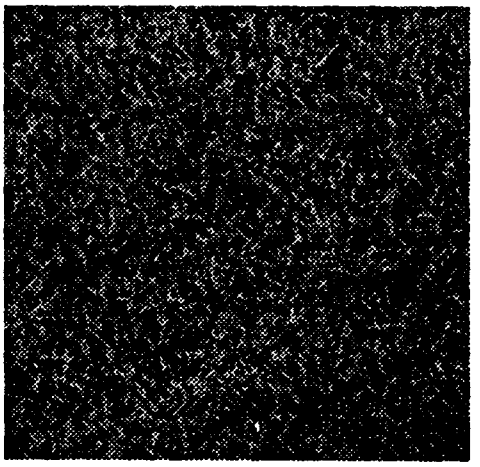
- Go to Step 4.

4 Experiments

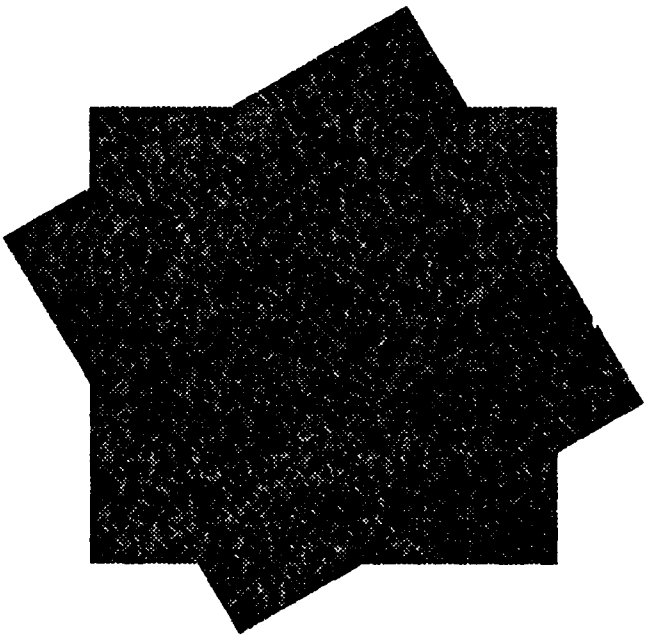
Texture Image Registration Registration of texture images is a difficult problem in computer vision. We have tested our algorithm on images of grass, leather, pigskin, sand, wood, and wool. Two typical results are presented here. In these experiments, the input pair of images are 512 × 512, obtained by digitizing a photo [3, 19] and its rotated version. The angles of rotation are about 30° as measured by a goniometer. So the true transform between the texture image pairs is expected to have a rotation of about 30°, scale close to 1, and a small amount of translation. In the implementation of our registration algorithm, we let the image size for the lowest resolution layer be 128 × 128 and the m and n of (4) be 2 and 5 respectively. Only feature points which are maximum for a radius of $r = \alpha^n$ are selected. The matching window parameter and the search space parameter are set to $\omega_m = 8$ and $\omega_s = 3$ respectively. In our implementation, matches with correlation coefficient $\psi < \frac{3}{4}$ are considered insignificant and are not used in motion estimation, unless the matches are among the top two candidates with the highest correlation.



(2.a) Grass–00



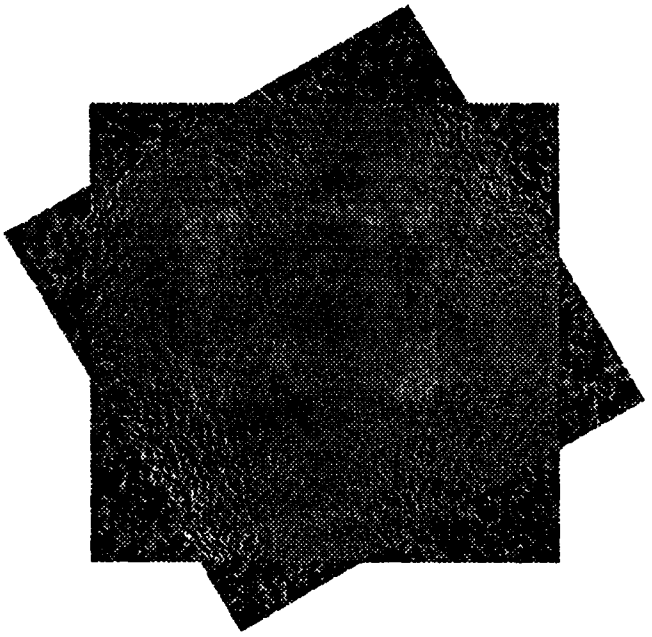
(2.b) Grass–30



(2.c) Mosaicking of Grass–00 and Grass–30

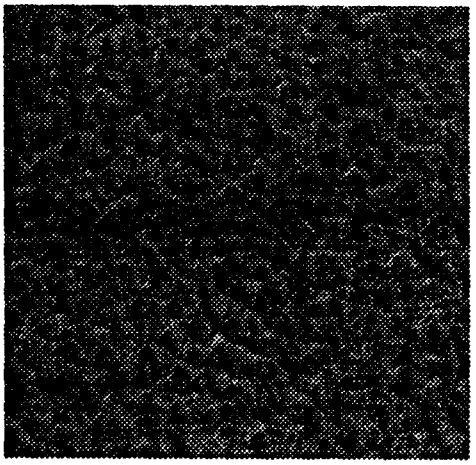
Figure 2 presents the registration of two pictures of a grassy field. (a) and (b) are the input images. (c) is the mosaicking of the transformed 0° picture and 30° picture. The transformation is done by the estimated motion parameters, which are $s = 1.00128$, $\theta = -30.2141°$, $\Delta X = 0.1374$ pixel, and $\Delta Y = 17.0161$ pixels.

Figure 3 presents the registration of two pictures of pigskin. (a) and (b) are the input images. (c) is the mosaicking of the transformed 0° picture and 30° picture. (d) is the difference between the transformed 0°

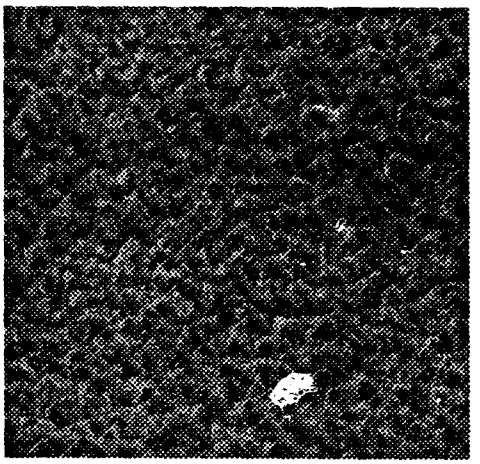


(2.d) Difference after registration

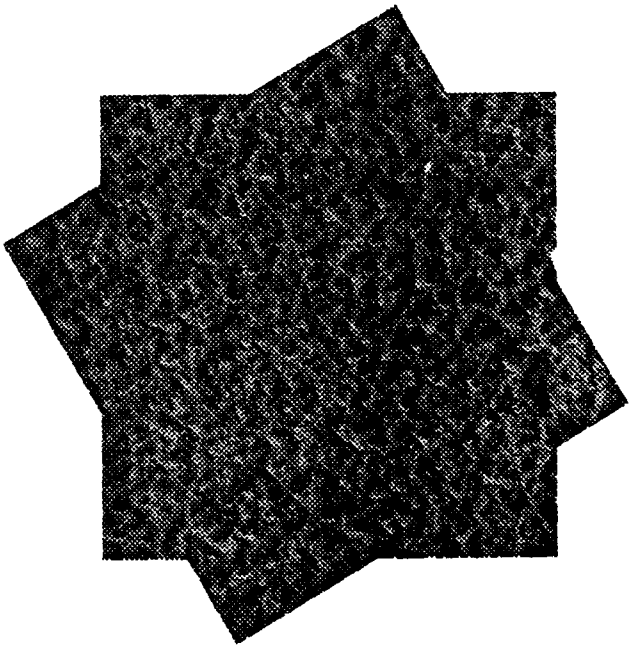
Figure 2: Registration of grass pictures. In (2.d) the zero of difference is shifted to 128.



(3.a) Pigskin-00



(3.b) Pigskin-30



(3.c) Mosaicking of Pigskin-00 and Pigskin-30

Figure 3: Registration of pigskin pictures.

picture and 30° picture. The transformation is done by the estimated motion parameters, which are $s = 1.00024$, $\theta = -30.6011°$, $\Delta X = 2.4026$ pixels, and $\Delta Y = 15.3286$ pixels.

In (d) of Figure 2, there are noticeable error rings in the central area. These are the errors generated by the digitizer. In spite of the digitization error, our algorithm produces correct matches. The continuity in the mosaicking image and the patterns in the difference image shows that the estimates are reasonably correct. Further quantitative error analysis can be performed by checking the consistency between forward and backward motion estimation. For image pair $f_1(X_1, Y_1)$ and $f_2(X_2, Y_2)$, assume that the estimated parameters of motion from $f_1(X_1, Y_1)$ to $f_2(X_2, Y_2)$ are s_1, θ_1, ΔX_1, and ΔY_1, and the estimated parameters of motion from $f_2(X_2, Y_2)$ to $f_1(X_2, Y_2)$ are s_2, θ_2, ΔX_2, and ΔY_2; then we have

$$\begin{pmatrix} X_2 \\ Y_2 \end{pmatrix} = s_1 \begin{pmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{pmatrix} \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} + \begin{pmatrix} \Delta X_1 \\ \Delta Y_1 \end{pmatrix} \quad (18)$$

$$\begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} = s_2 \begin{pmatrix} \cos\theta_2 & \sin\theta_2 \\ -\sin\theta_2 & \cos\theta_2 \end{pmatrix} \begin{pmatrix} X_2 \\ Y_2 \end{pmatrix} + \begin{pmatrix} \Delta X_2 \\ \Delta Y_2 \end{pmatrix} \quad (19)$$

Combining (18) and (19) leads to

$$\begin{aligned} \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} &= s_1 s_2 \begin{pmatrix} \cos(\theta_1+\theta_2) & \sin(\theta_1+\theta_2) \\ -\sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) \end{pmatrix} \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} \\ &\quad + s_2 \begin{pmatrix} \cos\theta_2 & \sin\theta_2 \\ -\sin\theta_2 & \cos\theta_2 \end{pmatrix} \begin{pmatrix} \Delta X_1 \\ \Delta Y_1 \end{pmatrix} + \begin{pmatrix} \Delta X_2 \\ \Delta Y_2 \end{pmatrix} \\ \begin{pmatrix} X_2 \\ Y_2 \end{pmatrix} &= s_1 s_2 \begin{pmatrix} \cos(\theta_1+\theta_2) & \sin(\theta_1+\theta_2) \\ -\sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) \end{pmatrix} \begin{pmatrix} X_2 \\ Y_2 \end{pmatrix} \\ &\quad + s_1 \begin{pmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{pmatrix} \begin{pmatrix} \Delta X_2 \\ \Delta Y_2 \end{pmatrix} + \begin{pmatrix} \Delta X_1 \\ \Delta Y_1 \end{pmatrix} \end{aligned}$$

The differences between the forward and backward estimates of (s, θ, ΔX, ΔY) can be defined as

$$\epsilon_s = |s_1 s_2 - 1| \quad (20)$$

$$\epsilon_\theta = |\theta_1 + \theta_2| \quad (21)$$

$$\epsilon_{\Delta X_f} = s_2(\cos\theta_2 \Delta X_1 + \sin\theta_2 \Delta Y_1) + \Delta X_2 \quad (22)$$

$$\epsilon_{\Delta Y_f} = s_2(-\sin\theta_2 \Delta X_1 + \cos\theta_2 \Delta Y_1) + \Delta Y_2 \quad (23)$$

$$\epsilon_{\Delta X_b} = s_1(\cos\theta_1 \Delta X_2 + \sin\theta_1 \Delta Y_2) + \Delta X_1 \quad (24)$$

$$\epsilon_{\Delta Y_b} = s_1(-\sin\theta_1 \Delta X_2 + \cos\theta_1 \Delta Y_2) + \Delta Y_1 \quad (25)$$

In all our experiments on texture image registration, the differences between the forward and backward estimates are bounded by

$$\begin{aligned} \epsilon_s &< 2 \times 10^{-3} \\ \epsilon_\theta &< 0.07° \\ \epsilon_\Delta &< 0.5 \text{ pixel} \end{aligned}$$

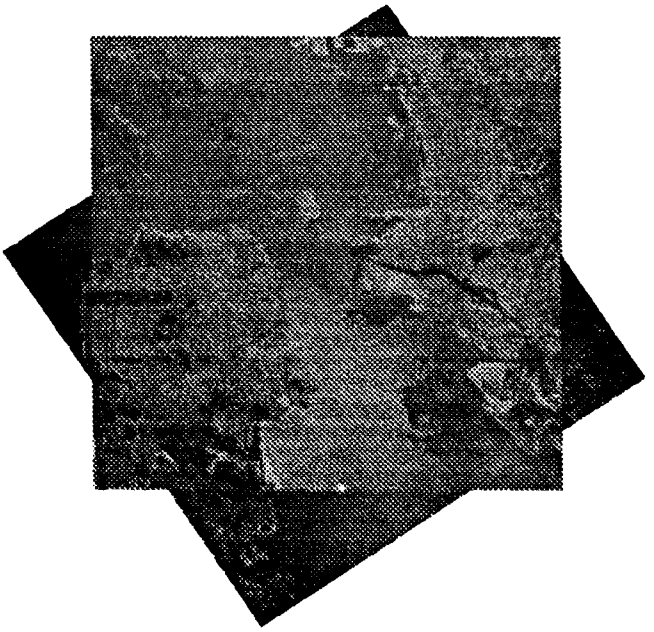
Matching of Aerial Images Automatic image registration is an important issue in remote sensing applications. Figures 4 and 5 show the results of using the registration algorithm for registering satellite images. Figure 4 shows the registration of two San Francisco images with significant differences in image orientations and intensities. SF3212 and SF3222, shown in



(4.1) SF3212



(4.b) SF3222



(4.c) Mosaicking of (4.a) and (4.b)

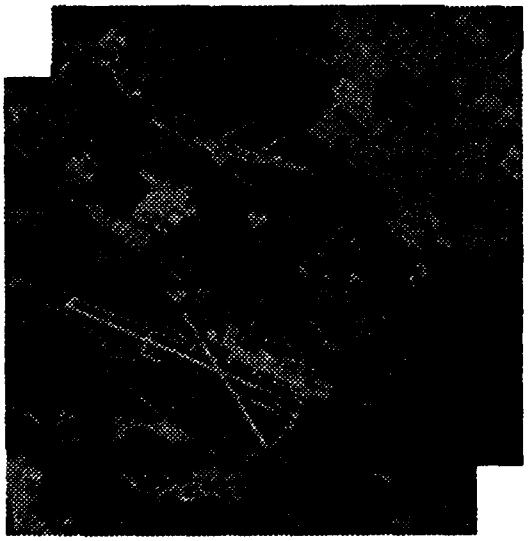
Figure 4: Registration of the San Francisco images.



(5.a) SD311



(5.b) SD325



(5.c) Mosaicking of (5.a) and (5.b)

Figure 5: Registration of the San Diego images.



(6.a) frame–1

(a) and (b), are the input images. The mosaicking of the transformed SF3212 and SF3222 is shown in (c). The estimated transform parameters are $s = 0.983908$, $\theta = 56.3428°$, $\Delta X = -3.17038$, and $\Delta Y = -57.8287$.

Figure 5 shows the registration of two San Diego images, in which (a) and (b) are the input images. The mosaicking of the transformed SD311 and SD325 is shown in (c). The estimated transform parameters are $s = 0.999911$, $\theta = 0.222798°$, $\Delta X = 51.2238$, and $\Delta Y = 78.6986$. In Figures (5.c) inspection of the continuity of features in the mosaicking image shows that the registering is correct.

Matching of Stereo Images Most of the known stereo algorithms assume that the input images have already been aligned so that the epipolar line is parallel to the scanning direction; also the images are roughly registered and the scaling between the image pair is adjusted. In many situations, this initial matching is performed by manually picking some feature points and aligning the images using a stereoscopic platform. Our registration algorithm is useful for obtaining the initial matching and the direction of the epipolar lines. Two experimental results on stereo image pair registration are presented here.

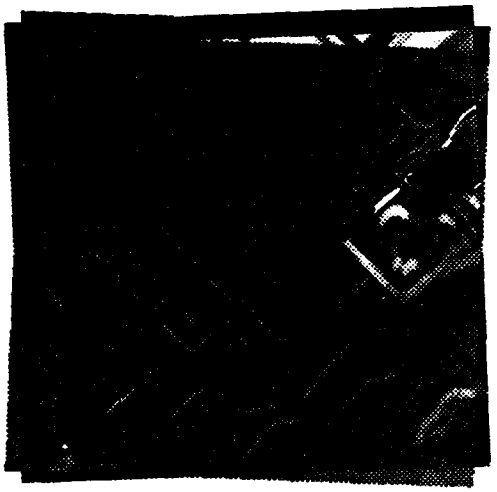
Figure 6 shows results on matching the first two frames of a robot arm image sequence[2]. (a) and (b) show the input images. (c) shows the difference between the transformed (a) and (b). The estimated transform parameters are $s = 1.00131$, $\theta = 4.19226°$, $\Delta X = -0.125131$, and $\Delta Y = -5.25858$.

Figure 7 shows an experiment on matching the first and last frames of a chemical plant image sequence (close view). In Figure 7, (a) and (b) are the two input frames of the chemical plant image sequence. (c) shows the difference between the motion compensated (a) and (b). The estimated motion parameters are $s = 1.06916$, $\theta = 0.175411°$, $\Delta X = -0.784010$, and $\Delta Y = -28.5614$. Note that the field of the last frame is smaller than the

[2]The robot arm image sequence was provided by University of Massachusetts, Department of Computer and Information Sciences.

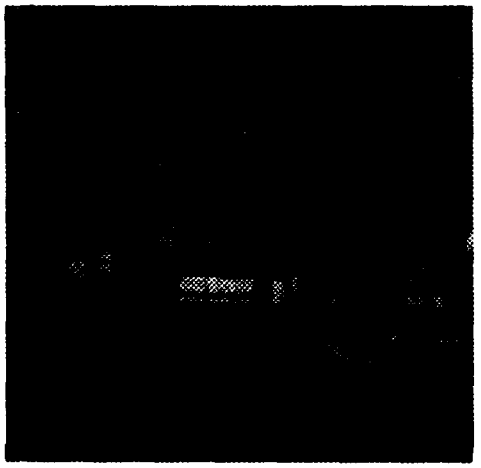


(6.b) frame–2



(6.c) Difference after registration

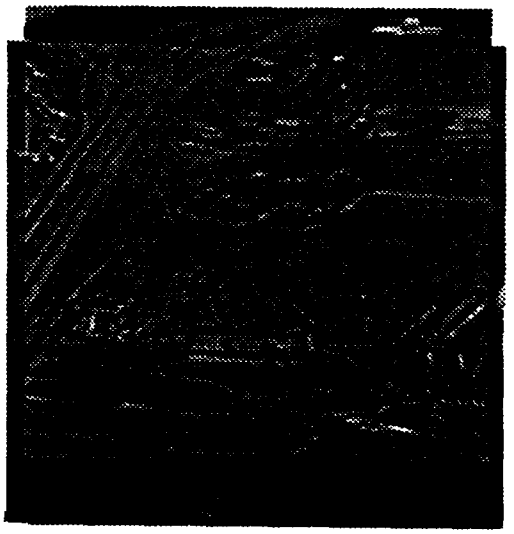
Figure 6: Registration of a robot arm image sequence (the first two frames). In (6.c) the first frame is first transformed using the estimated motion parameters.



(7.a) frame–1



(7.b) frame–32



(7.c) Difference after registration

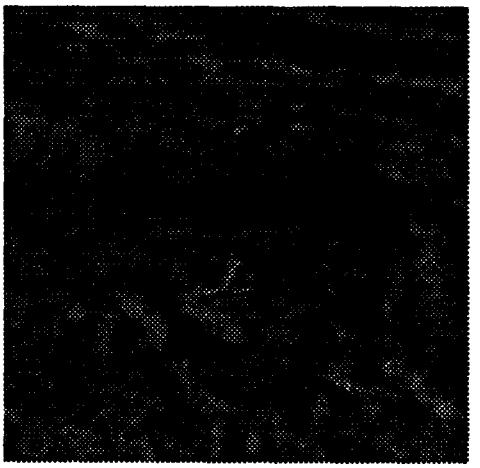
Figure 7: Registration of a chemical plant image sequence (close view). In (7.c) the first frame is first transformed using the estimated motion parameters.



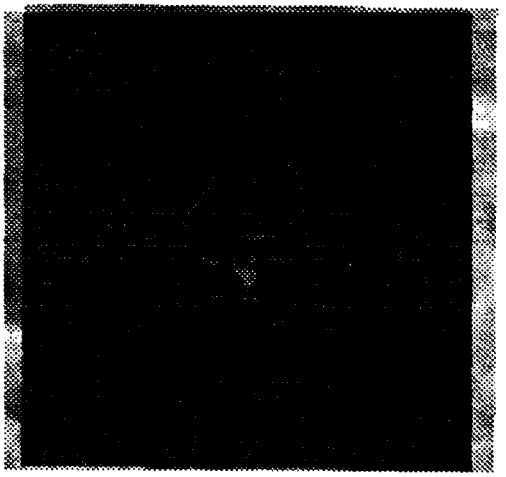
(8.a) first frame



(8.b) eighth frame



(8.c) Direct difference



(8.d) Motion compensated difference

Figure 8: Change detection from helicopter images

field of the first frame and the translation between the two frames is mainly in the vertical direction. These are consistent with the fact that the camera is approaching the chemical plant.

Moving Object Detection Our algorithm can also be used for change detection from satellite images. Here we present applications of our algorithm to moving object detection.

Figure 8 shows an experiment in change detection using the first and the eighth frames of a helicopter image sequence. The basic purpose is to detect the differences in successive images. To carry this out, the motion of the camera needs be compensated first. Our registration algorithm is useful in this application as demonstrated in the following. In Figure 8, (a) and (b) are the input images, (c) shows the direct difference between (a) and (b), and (d) shows the difference between (a) and (b) after the motion of the camera has been compensated. The improvement of (d) over (c) is obvious. The moving helicopter can be detected by thresholding the difference image shown in Figure (8.d).

5 Conclusion

A f[illegible] robust 2-D translation, rotation, and scaling [illegible]n algorithm has been presented. We have illu[illegible]ie performance of the algorithm on a variety [illegible]ages. Satisfactory registration results have been obtained for registration of texture images, stereo matching, satellite image mosaicking, and moving object detection.

ACKNOWLEDGMENT

The authors are thankful to Professor Azriel Rosenfeld of the University of Maryland for his comments on this report.

References

[1] P. E. Anuta, "Spatial Registration of Multispectral and Multitemporal Digital Imagery Using FFT Techniques," *IEEE Trans. Geoscience and Electronics*, Vol. GE-8, pp. 353–368, Oct. 1970.

[2] J. Bolz and C. D. Gilbert, "Generation of End-Inhibition in the Visual Cortex via Interlaminar Connections," *Nature*, Vol. 320, pp. 362–365, Mar. 1986.

[3] P. Brodatz, *Textures: A Photographic Album for Artists and Designers*, New York: Dover Publications, 1966.

[4] M. J. Brooks and B. K. P. Horn, "Shape and Source from Shading," in *Proc. Int. Joint Conf. Artificial Intell.*, (Los Angeles), pp. 932–936, 1985.

[5] R. T. Frankot, "'SAR Image Registration by Multi-Resolution Correlation," in *Proc. SPIE*, vol. 220, pp. 195–203, 1983.

[6] B. K. P. Horn, *Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View*, Ph.D. dissertation, Department of Electrical Engineering, MIT, 1970.

[7] B. K. P. Horn, "Obtaining Shape from Shading Information," in *The Psychology of Machine Vision* (P. H. Winston, ed.), pp. 115–155, New York: McGraw-Hill, 1975.

[8] B. K. P. Horn, "Height and Gradient from Shading," *Int. J. Computer Vision*, Vol. 5, pp. 584–595, Aug. 1990.

[9] B. K. P. Horn and M. J. Brooks, *Shape From Shading*, Cambridge, Massachusetts: MIT Press, 1989.

[10] T. S. Huang, *Image Sequence Analysis*, Berlin/Heidelberg: Springer-Verlag, 1981.

[11] D. H. Hubel and T. N. Wiesel, "Receptive Fields and Functional Architecture in Two Nonstriate Visual Areas(18 and 19) of the Cat," *J. Neurophysiology*, Vol. 28, pp. 229–289, Mar. 1965.

[12] C. D. Kuglin *et al.*, "Map Matching Techniques for Terminal Guidance Using Fourier Phase Information," in *Proc. SPIE*, vol. 186, 1979.

[13] C. D. Kuglin and D. C. Hines, "The Phase Correlation Image Alignment Method," in *IEEE Int. Conf. on Cybernetics and Society*, pp. 163–165, 1975.

[14] C. H. Lee and A. Rosenfeld, "Improved Methods of Estimating Shape from Shading Using the Light Source Coordinate System," in *Shape from Shading* (B. K. P. Horn and M. J. Brooks, eds.), pp. 323–569, MIT Press, 1989.

[15] B. S. Manjunath, R. Chellappa, and C. Malsburg, "Shape Representation and Identification: Application to Human Face Recognition." Working paper, 1991.

[16] A. Pentland, "Shape Information from Shading: A Theory About Human Perception," in *Proc. Int. Conf. Comput. Vision*, (Tarpon Springs, FL), pp. 404–413, 1988.

[17] A. P. Pentland, "Local Shading Analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-16, pp. 170–187, Mar. 1984.

[18] Q. Tian and M. N. Huhns, "Algorithms for Subpixel Registration," *Comput. Vision, Graphics, Image Processing*, Vol. 35, pp. 220–233, Aug. 1986.

[19] A. G. Weber, "Image Data Base," Tech. Rep. USC-SIPI 101, University of Southern California, Los Angeles, 1989.

[20] Q. Zheng and R. Chellappa, "Estimation of Illuminant Direction, Albedo and Shape from Shading," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-13, pp. 680–702, July 1991.

CALIBRATION OF CAMERAS USING THE ESSENTIAL MATRIX

Richard Hartley

G.E. CRD, Room K1-5C39
P.O. Box 8, Schenectady,
NY, 12301.

This paper considers the relative placement problem for two cameras viewing a point set in three dimensions. A non-iterative algorithm to solve this problem was given by Longuet-Higgins [3]. However, Longuet-Higgins's solution made certain assumptions about the camera that may not be justified in practice. In particular, it is assumed implicitly in his paper that the focal length of the two cameras is known, as is the principal point (the point where the focal axis of the camera intersects the image plane). Whereas it is often a safe assumption that the principal point of an image is at the center pixel, the focal length of the camera is not easily deduced, and will generally be unknown for images of unknown origin. In this paper, a non-iterative algorithm is given for the solution of the relative camera placement problem when the focal lengths of the cameras are unknown. It will also reexamine the algorithm of Longuet-Higgins and suggest a simplification.

The focal length of a camera determines the field of view of an image, or more particularly, the angle subtended per unit length in the focal plane. The effect of increased focal length is however indistinguishable from enlargement. For digital images digitized from film and sampled into discrete pixels, the concept of focal length is meaningless, unless the exact pixel size is known. For this reason, I prefer to use the term "magnification" in place of the term focal length.

1. THE 8-POINT ALGORITHM

First, I will derive the 8-point algorithm of Longuet-Higgins in order to fix notation and to gain some insight into its properties. An alternative derivation was given in [3] or [4]. Since we are dealing with homogeneous coordinates, we are interested only in values determined up to scale. Consequently we introduce the notation $\sim$ to indicate equality up to multiplication by a scale factor. Image space coordinates will usually be given in homogeneous coordinates as $(u, v, w)^T$.

1.1 Algorithm Derivation.

We consider the case of two cameras, one of which is situated at the origin $(0, 0, 0)^T$ of object space coordinates, and one which is displaced from it. The two cameras may be represented by the transformation that they perform translating points from object space into image space coordinates. The two transformations are assumed to be

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad \text{(Eq.1)}$$

and

$$\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = R\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix} - \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}\right) \qquad \text{(Eq. 2)}$$

where R is a rotation matrix, and the vectors $(u, v, w)^T$ and $(u', v', w')^T$ are the homogeneous coordinates of the image points. Writing $T = (t_x, t_y, t_z)^T$, and using homogeneous coordinates in both object and image space, the above relations may be written in matrix form as

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = (I \mid 0)\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = P_1\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad \text{(Eq. 3)}$$

and

$$\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = (R \mid -RT)\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = P_2\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad \text{(Eq. 4)}$$

where $(R \mid -RT)$ and $(I \mid 0)$ are 3×4 matrices divided into a 3×3 block and a 3×1 column and I is the identity matrix.

Now, I will define a transformation between the 2-dimensional projective plane of image coordinates in image 1 and the pencil of epipolar lines in the second image. In particular, as is well known, given a point $(u, v, w)^T$ in image 1, the corresponding point in image 2 must lie on a certain epipolar line, which is the image under P_2 of the set S of all points $(x, y, z, 1)^T$ which map under P_1 to $(u, v, w)^T$. To determine this line one may identify two points in S, namely the camera origin $(0, 0, 0, 1)^T$ and the point at infinity, $(u, v, w, 0)^T$. The images of these two points under P_2 are $-RT$ and $R(u, v, w)^T$ respectively. Consequently, the line that passes through these two points is given in homogeneous coordinates by their cross product,

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = RT \times R\begin{pmatrix} u \\ v \\ w \end{pmatrix} = R.\left(T \times \begin{pmatrix} u \\ v \\ w \end{pmatrix}\right) \qquad \text{(Eq. 5)}$$

Here $(p, q, r)^T$ represents the line $pu' + qv' + rw' = 0$. Representing by S the matrix

$$S = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \qquad \text{(Eq. 6)}$$

equation (Eq. 5) may be written as

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = RS \begin{pmatrix} u \\ v \\ w \end{pmatrix} \qquad \text{(Eq. 7)}$$

Since the point $(u', v', w')^T$ corresponding to $(u, v, w)^T$ must lie on the epipolar line, we have the important relation

$$(u', v', w')\, Q\, (u, v, w)^T = 0 \qquad \text{(Eq. 8)}$$

where Q = RS. This relationship is due to Longuet-Higgins ([3]).

As is well known, given 8 correspondences of more, the matrix Q may be computed by solving a (possibly overdetermined) set of linear equations. In order to compute the second camera transform, P_2, it is necessary to factor Q into the product RS of a rotation matrix and a skew-symmetric matrix. Longuet-Higgins [3] gives a rather involved, and apparently numerically somewhat unstable method of doing this. I will give an alternative method of factoring the Q matrix based on the Singular Value Decomposition ([5]). The following result may be verified.

Theorem 1: A 3×3 matrix Q can be factored as the product of a rotation matrix and a non-zero skew symmetric matrix if and only if Q has two equal non-zero singular values and one singular value equal to 0. //

For a proof see [1].

This theorem allows us to give an easy method of factoring any matrix into a product RS, where possible.

Theorem 2: Suppose the matrix Q can be factored into a product RS where R is orthogonal and S is skew-symmetric. Let the singular value decomposition of Q be UDV^T where D = diag (k, k, 0). Then up to a scale factor the factorization is one of the following :

$$S \sim VZV^T \; ; \; R \sim UEV^T \text{ or } UE^TV^T \; ;$$
$$Q \sim RS.$$

where

$$E = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \; Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \text{(Eq. 9)}$$

//

Proof : That the given factorization is valid is true by inspection. That these are the only solutions is implicit in the paper of Longuet-Higgins. //

It may be verified that T (the translation vector) in Theorem 2 is equal to $V.(0,0,1)^T$ since this ensures that ST = 0 as required by (Eq. 6). Furthermore $\| T \| = 1$, which is a convenient normalization suggested in [3]. As remarked by Longuet-Higgins, the correct solution to the camera placement problem may be chosen based on the requirement that the visible points be in front of both cameras ([3]). There are four possible rotation-translation pairs that must be considered based on the two possible choices of R and two possible signs of T. Therefore, since $UEV^T.V(0, 0, 1)^T = U(0, 0, 1)^T$ the requisite camera matrix $(R \mid -RT)$ is equal to $(UEV^T \mid U(0, 0, 1)^T)$ or one of the obvious alternatives.

1.2. Numerical Considerations.

In any practical application, the matrix Q found will not factor exactly in the required manner because of inaccuracies of measurement. In this case, the requirement will be to find the matrix "closest" to Q that does factor into a product RS. Using the sum of squares of matrix entries as a norm, we wish to find the matrix Q' = RS such that $\| Q - Q' \|$ is minimized. The following theorem shows that the factorization given in the previous theorem is numerically optimal.

Theorem : Let Q be any 3 by 3 matrix and $Q = UDV^T$ be its singular value decomposition in which D = diag (r, s, t) and $r \geq s \geq t$. Define the matrix Q' by $Q' = U\, D'\, V^T$ where D' = diag (k, k, 0) and k = (r+s)/2. Then Q' is the matrix closest to Q under the sum-of-squares norm which satisfies the condition Q' = R.S, where R is a rotation and S is skew-symmetric. Furthermore, the factorization is given up to sign by $R = UEV^T$ or UE^TV^T and $S = k.VZV^T$. //

This theorem is plausible given the norm-preserving property of orthogonal transformations. However, its proof is not entirely obvious and falls outside of the scope of this paper.

1.3. Algorithm Outline.

The algorithm for computing relative camera locations for calibrated cameras is as follows.

1. Find Q by solving a set of equations of the form (Eq. 8)
2. Find the singular value decomposition Q = UDV^T, where D = diag(a, b, c) and $a \geq b \geq c$.
3. The transformation matrices for the two cameras are $C_1 = (I \mid 0)$ and C2 equal to one of the four following matrices.

$$(UEV^T \mid U(0, 0, 1)^T)$$
$$(UEV^T \mid U(0, 0, -1)^T)$$
$$(UE^TV^T \mid U(0, 0, 1)^T)$$
$$(UE^TV^T \mid U(0, 0, -1)^T)$$

The choice between the four transformations for C2 is determined by the requirement that the point locations (which may be computed once the cameras are known [3]) must lie in front of both cameras. Geometrically, the camera rotations represented by UEV^T and UE^TV^T differ from each other by a rotation through 180 degrees about the line joining the two cameras. Given this fact, it may be verified geometrically that a single pixel-to-pixel correspondence is enough to eliminate all but one of the four alternative camera placements.

2. UNCALIBRATED CAMERAS.

If the internal camera calibration is not known, then the problem of finding the camera parameters is considerably more difficult. In general one would like to allow arbitrary non-singular matrices K describing internal camera calibration and consider camera matrices of the general form $(K_2R \mid -K_2RT)$, that is, general 3×4 matrices. Because K is multiplied by a rotation, R, it may be assumed that K is upper triangular. Further allowing for an arbitrary scale factor, there are 5 remaining independent entries in K representing camera parameters. Other authors have allowed four internal camera parameters, namely principal point offsets in two directions and different scale factors in two directions. If however different scaling is allowed in two directions not necessarily aligned with the direction of the image-space axes, then one more parameter is needed, *making up the 5.*

It is too much to hope that from a set of image point correspondences one could hope to retrieve the full set of internal camera parameters for a pair of cameras as well as the relative external positioning of the cameras. Indeed if $\{x_i\}$ are a set of points visible in a pair of cameras with transform matrices M_1 and M_2, and G is an arbitrary non-singular 4×4 matrix, then replacing each x_i by $G^{-1}x_i$ and each camera M_j with M_jG preserves the object-point to image-space correspondences. As may be seen, the internal parameters of one of the cameras, M_1 say, may be chosen arbitrarily. The situation is not helped by adding more cameras. This is in contrast to the case of calibrated cameras in which a finite number of solutions are possible ([1]) The question remains, therefore, how much can be deduced about the internal camera parameters from a set of image correspondences. It will be shown in this paper that assuming all other internal parameters to be known, the magnification factors of the two cameras may be computed.

2.1 Algorithm Derivation

Let K_1 and K_2 be two matrices representing the internal camera transformations of the two cameras and let $P_1 = (K_1 \mid 0)$ and $P_2 = (K_2R \mid -K_2RT)$ be the two camera transforms. The task is to obtain R, T, K_1 and K_2 given a set of image-point correspondences. For the present, the matrices K_1 and K_2 will be assumed arbitrary.

As before, it is possible to determine the epipolar line corresponding to a point $(u, v, w)^T$ in image 1. The two points that must lie on the epipolar line are the images under P_2 of the camera centre $(0, 0, 0, 1)^T$ of the first camera and the point at infinity $\begin{pmatrix} K_1^{-1}\begin{pmatrix} u \\ v \\ w \end{pmatrix} \\ 0 \end{pmatrix}$. Transform P_2 takes these two points to the points $-K_2RT$ and $K_2RK_1^{-1}.(u, v, w)^T$. The line through these these points is given by the cross product

$$K_2RT \times K_2RK_1^{-1}(u, v, w)^T. \qquad \text{(Eq. 10)}$$

If K is a square matrix, we use the notation K* to represent the adjoint of K, that is the matrix of cofactors defined by $K^*_{ij} = (-1)^{i+j}\det(K^{(ij)})$ where $K^{(ij)}$ is the matrix derived from K by removing the i-th row and j-th column. If K is non-singular, then it is well known that $K^* = \det(K) \cdot (K^T)^{-1}$. In other words, $K^* \approx (K^T)^{-1}$ The adjoint matrix is related to cross products in the following way.

Remark : If **a** and **b** are 3-dimensional column vectors and K is a 3×3 matrix, then $K\mathbf{a} \times K\mathbf{b} \approx K^*(\mathbf{a} \times \mathbf{b})$.

Using this fact it is easy to evaluate the cross product

$$K_2RT \times K_2RK_1^{-1}(u, v, w)^T \approx K_2^*RK_1^{*-1}(K_1T \times (u, v, w)^T) \qquad \text{(Eq. 11)}$$

If $(t_x', t_y', t_z')^T = K_1T$ then writing

$$S = \begin{pmatrix} 0 & -t_z' & t_y' \\ t_z' & 0 & -t_x' \\ -t_y' & t_x' & 0 \end{pmatrix} \qquad \text{(Eq. 12)}$$

as before, we have a formula for the epipolar line corresponding to the point $(u, v, w)^T$ in image 1.

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} \approx K_2^*RK_1^TS\begin{pmatrix} u \\ v \\ w \end{pmatrix} \qquad \text{(Eq. 13)}$$

Furthermore, setting $Q = K_2^{-1}RK_1S$ we have the formula

$$(u', v', w')\,Q\,(u, v, w)^T = 0. \qquad \text{(Eq. 14)}$$

After solving to find the matrix Q as before, we are faced with the more difficult task of finding the factorization $Q \approx K_2^*RK_1^TS$.

An alternative factorization for Q that may be derived from (Eq. 10) is

$$Q \approx (K_2^{-1})^TRSK_1^{-1} \qquad \text{(Eq. 15)}$$

where S' is as given in (Eq. 6).

2.2. Factorization of Q.

Our goal is to find the factorization $Q \approx K_2^*RK_1^TS$. As before, we use the Singular Value Decomposition, $Q = UDW^T$. Since Q is non-singular, the diagonal matrix D equals diag (r, s, 0) where r and s are two positive constants.

Since $QW(0, 0, 1)^T = 0$, it follows that $SW(0, 0, 1)^T = 0$ since $K_2{*}RK_1{}^T$ is non-singular, and so

$$S \sim \left(W \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} W^T \right)$$

as in section 1.2. The general solution to the problem of factoring Q into a product R'S, where R' is non-singular and S is skew-symmetric is therefore given by

$$Q = \left(U \begin{pmatrix} r & 0 & \alpha \\ 0 & s & \beta \\ 0 & 0 & \gamma \end{pmatrix} E^T W^T \right) \cdot \left(W \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} W^T \right)$$

(Eq. 16)

where α, β and γ are arbitrary constants. The two bracketed expressions are R' and S respectively and the factorization is unique (except for the variables α, β and γ) up to scale. Unlike in the case in section 1.2, we do not need to consider the alternate solution in which E^T is replaced by E, since that is taken care of by the undetermined values α, β and γ. Since both E and W are orthogonal matrices, we write V = WE, and V is also orthogonal. Further writing

$$X_{\alpha,\beta,\gamma} = \begin{pmatrix} r & 0 & \alpha \\ 0 & s & \beta \\ 0 & 0 & \gamma \end{pmatrix} \qquad \text{(Eq. 17)}$$

we obtain the expression $R' = U X_{\alpha,\beta,\gamma} V^T$.

Now, we turn our attention to the matrix R'. For some values of α,β and γ, we must have an identity $R' \sim K_2{*}RK_1{}^{*-1}$, where R is a rotation matrix. From this it follows that $R \sim K_2{}^{*-1}R' K_1{*}$. The particular property of a rotation matrix that we will use is that it is equal to its adjoint, (inverse transpose). This means that $K_2{}^{*-1}R'K_1{*} \sim K_2{}^{-1} R' * K_1$ or

$$K_2K_2{}^T R' \sim R' * K_1K_1{}^T. \qquad \text{(Eq. 18)}$$

Since $R' = UXV^T$, it follows that $R'* \sim U X* V^T$ and X* is the matrix

$$X* = X*_{\alpha,\beta,\gamma} = \begin{pmatrix} s\gamma & 0 & 0 \\ 0 & r\gamma & 0 \\ -s\alpha & -r\beta & rs \end{pmatrix} \text{(Eq. 19)}$$

and

$$(K_2K_2{}^T)\, UX_{\alpha,\beta,\gamma}V^T \sim UX*_{\alpha,\beta,\gamma}V^T\, (K_1K_1{}^T)$$

(Eq. 20)

At this point, it is necessary to specialize to the case where K_1 and K_2 are of the simple form $K_1 = \text{diag}(1, 1, k_1)$ and $K_2 = \text{diag}(1, 1, k_2)$. In this case, k_1 and k_2 are the inverses of the magnification factors. If the entries of UXW^T are (f_{ij}) and those of $UX*W^T$ are (g_{ij}), then multiplying by $(K_2K_2{}^T)$ and $(K_1K_1{}^T)$ respectively gives an equation

$$\begin{pmatrix} f11 & f12 & f13 \\ f21 & f22 & f23 \\ k_2{}^2f31 & k_2{}^2f32 & k_2{}^2f33 \end{pmatrix} = x \begin{pmatrix} g11 & g12 & k_1{}^2g13 \\ g21 & g22 & k_1{}^2g23 \\ g31 & g32 & k_1{}^2g33 \end{pmatrix}$$

(Eq. 21)

where the f_{ij} and g_{ij} linear expressions in α, β and γ, and x is an unknown scale factor.

The top left hand block of this equation set comprises a set of equations of the form

$$\begin{pmatrix} f11 & f12 \\ f21 & f22 \end{pmatrix} = x \begin{pmatrix} g11 & g12 \\ g21 & g22 \end{pmatrix} \qquad \text{(Eq. 22)}$$

If the scale factor were known, then this system could be solved as a set of linear equations. Unfortunately, x is not known, and it is necessary to find the value of x before solving the set of linear equations for α, β and γ.

Since the entries of the matrices on both sides of (Eq. 21) are linear expressions in α, β and γ, it is possible to rewrite (Eq. 22) in the form

$$M_1 \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ 1 \end{pmatrix} - x.\, M_X \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ 1 \end{pmatrix} = 0, \quad \text{(Eq. 23)}$$

where M_1 and M_X are 4×4 matrices, each row of M_1 or M_X corresponding to one of the four entries in the matrices in (Eq. 22). Such a set of equations has a solution only if $\det(M_1 - x.M_X) = 0$.

This leads to a polynomial equation of degree 4 in x : $p(x) = \det(M_1 - xM_X) = 0$. This equation may be solved to give a value of x. It turns out, however, that it is sufficient to solve a quadratic equation. In particular, it may be shown that $\det(M_1)$ and $\det(M_X)$ are both zero. Since these represent the constant and fourth-order terms of $\det(M_1+xM_X)$ it follows that $\det(M_1+xM_X)$ is a cubic polynomial with one root equal to 0. To solve the equation, therefore, it is necessary only to solve a quadratic equation.

At this stage we know that $\det(M_1 - x.M_X)$ is a polynomial of the form $p(x) = a_3x^3 + a_2x^2 + a_1x$ and so may be easily solved. The root $x = 0$ of this polynomial may safely be ignored, since according to (Eq. 22) it would imply that $f_{ij} = 0$ for $i, j \leq 2$, and hence that R' is singular, which by assumption it is not.

Once the value of x is known, the values of α, β and γ may be determined by solving the set of equations given in (Eq. 22). Finally, the values of k_1 and k_2 may be read off from equations (Eq. 21). In particular,

$$k_2^2 = x.g_{31}/f_{31} = x.g_{32}/f_{32}$$

$$k_1^2 = f_{13}/x.g_{13} = f_{23}/x.g_{23} \qquad \text{(Eq. 24)}$$

$$k_2^2 f_{33} = x.k_1^2 g_{33}.$$

This redundancy gives a check on the correctness of the computed values of k_1 and k_2 which may be used to distinguish between the two roots of p(x). In practice, however, several observations have been made which could be used to simplify the case checking.

1. The two roots of the polynomial $p(x) = \det(M_1 - xM_x)$ are negatives of each other.
2. For some Q matrices, the determinant p(x) has purely imaginary roots. This would imply the impossibility of a real solution and so could not occur if the Q matrix is derived from the problem of image-point matches.
3. The estimated value of k_1^2 and k_2^2 corresponding to the two opposite roots of p(x) are negatives of each other. In one of these cases, therefore, k_i^2 is negative, and may be ignored.
4. The two estimates for and k_2^2 given by the first line of (Eq. 27) are exactly equal. The same holds for the two values of k_1^2.

At this point, it is possible to continue and compute the values of the rotation matrix from (Eq. 17). However, it turns out to be more convenient, now that the values of the magnification are known, to revert to the case of a calibrated camera. More particularly, we observe that according to (Eq. 15), Q may be written as $Q = K_2^{-1} Q' K_1^{-1}$ where $Q' = RS$, and R is a rotation matrix. The original method of section 1.3 may now be used to solve for the camera matrices derived from Q'. In this way, we find camera models $C_1 = (I \mid 0)$ and $C_2 = (R \mid -RT)$ for the two cameras corresponding to Q'. Taking account of the magnification matrices K_1 and K_2, the final estimates of the camera matrices are $(K_1 \mid 0)$ and $(K_2R \mid -K_2RT)$.

In practice it has been observed that greater numerical accuracy is obtained by repeating the computation of k_1 and k_2 after replacing Q by Q'. The values of k_1 and k_2 computed from Q' are very close to 1 and may be used to revise the computed magnifications very slightly. However, such a revision is necessary only because of numerical round-off error in the algorithm and is not strictly necessary.

3. PRACTICAL RESULTS.

This algorithm has been encoded in C and tested on a variety of examples. In the first test, a set of 25 matched points was computed synthetically, corresponding to an oblique placement of two cameras with equal magnification values of 1003. The principal point offset was assumed known. The solution to the relative camera placement problem was computed. The two cameras were computed to have magnifications of 1003.52 and 1003.71, very close to the original. Camera placements and point positions were computed and were found to match the input pixel position data within limits of accuracy. Similarly, the positions in 3-space of the object points matched the known positions to within one part in 10^4.

The algorithm was also tested out on a set of matched points derived from a stereo-matching program, STEREOSYS ([2]). A set of 124 matched points were found by an unconstrained hierarchical search. The two images used were 1024 by 1024 aerial overhead images of the Malibu region with about 40% overlap. The algorithm described here was applied to the set of 124 matched points and relative camera placements and object-point positions were computed. The computed model was then evaluated against the original data. Consequently, the computed camera models were applied to the computed 3-D object points to give new pixel locations which were then compared with the original reference pixel data. The RMS pixel error was found to be 0.11 pixels. In other words, the derived model matches the actual data with a standard deviation of 0.11 pixels. This shows the accuracy not only of the described camera modelling data, but also the accuracy of the point-matching algorithms.

REFERENCES.

1. Olivier Faugeras and Steve Maybank, "Motion from Point Matches : Multiplicity of Solutions," International Journal of Computer Vision, 4, 225-246 (1990).

2. M. J. Hanna "Bootstrap Stereo," Proc. Image Understanding Workshop, College Park, MD, April 1980, pp. 201-208.

3. H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections", Nature, Vol 293, 10 Sept. 1981.

4. John Porrill and Stephen Pollard, "Curve Fitting and Stereo Calibration", Proceedings of the British Machine Vision Conference, University of Oxford, Sept 1990, pp 37-42.

5. Kendall E. Atkinson "An Introduction to Numerical Analysis", John Wiley and Sons, second edition 1989.

Single Plane Model Extension using Projective Transformations

Robert T. Collins

Computer and Information Science Department
University of Massachusetts at Amherst
Amherst, MA. 01003

Abstract

It is well known that given at least four reference points/lines on a planar surface and their correspondences in the image, the relative positions of other points/lines on the plane can be derived regardless of camera position or intrinsic calibration parameters. This result is used to perform model extension for points and lines on planar surfaces. A new framework for data fusion in the projective plane is presented to merge the relative positions of coplanar points and lines derived in this way.

1 Planar Model Extension

Acquiring 3D models of the environment is an important current research problem in computer vision. Modeling the world in all its complexity is a daunting task, due to open issues in representing curved surfaces and volumes, and the variety of textures found in natural scenes. Because of these difficulties, many researchers have focused on man-made domains where planar surfaces and linear surface markings predominate. The limitations of these domains are mitigated by the fact that useful applications exist where the planarity assumption does hold, such as indoor mobile robot navigation. Even in unrestricted environments, the world is sometimes flat enough locally to approximate by piecewise planar patches.

The major benefit of adopting the world-planarity assumption is that the relevant geometric entities, namely points, lines and planes, are easily represented as linear subspaces. Furthermore, a rich set of results from the field of projective geometry become available. The relevance of projective geometry to the visual aquisition of planar surface models cannot be overstressed: *Projective geometry provides a mathematical foundation for characterizing and representing the relationships between linear subspaces that remain invariant under the imaging process.*

*This work was funded by DARPA and TACOM under contract number DAAE07-91-C-R035 and by the National Science Foundation under grant number CDA-8922572.

This paper describes an approach to model extension using properties of projective mappings between planes. Model extension is just one application of a general framework being developed for geometric inference in projective space. The term model extension reflects the notion of combining apriori structure information from a partial model with the observed structure of an image, to derive further extensions to the model. In particular, apriori knowledge of the relative positions of at least four coplanar points or lines is used to derive the positions of other points and lines on the same plane in a way that is invariant to relative camera location and intrinsic camera parameters. One of the main contributions of this work is the development of an appropriate methodology for fusing geometric information in the projective plane.

Section 2 presents background material showing that a number of useful plane to plane correspondences can be described by invertible linear transformations called *homographies*. The condition of invertibility is crucial to enabling geometric inference between planes, but it requires that the planes be treated as projective, rather than affine or Euclidean, a distinction that becomes important when considering how to represent and propagate uncertainty in the data. In Section 3, an approach to single plane model extension using homographies is described that decouples determination of the relative 2D locations of coplanar points and lines from the calculation of the 3D position of the plane in which they lie, allowing reconstruction of a planar object face without knowing the position of the camera or the appropriate camera calibration parameters. In Section 4 geometric information derived from multiple views is fused in the projective plane using an antipodally symmetric probability distribution on the unit sphere.

2 Projective Transformations

This section briefly summarizes properties of projective mappings between planes and their representation as homographies. Some of the relevant material can be found in [Mohr91, Faug88, Tsai82]. For

a more comprehensive discussion of projective transformations, the reader is invited to consult a projective geometry text such as [Spri64].

2.1 Homographies

A general projective transformation between planes can be written algebraically as

$$X' = \frac{aX + bY + c}{gX + hY + i} \quad , \quad Y' = \frac{dX + eY + f}{gX + hY + i} \qquad (1)$$

where (X, Y) and (X', Y') are points represented in the 2D local coordinate systems of each plane. Unfortunately, this is a nonlinear transformation that is undefined when the denominator is zero, corresponding to a point mapping to infinity.

In order to make a projective transformation bijective, a line of points at infinity is explicitly added to each plane, to correspond to the cases where the denominator in (1) goes to zero. A plane that has been augmented in this way is a new geometric entity called the *projective plane.* The projective plane has a different global topology than the affine or Euclidean plane, and this has implications for the representation of observed points and their uncertainty. This topic is explored in Section 4.

Using homogeneous coordinates, infinite points can be manipulated the same as finite ones, and the transformation of equation (1) become linear

$$k \begin{bmatrix} X' \\ Y' \\ S' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} X \\ Y \\ S \end{bmatrix}, \qquad (2)$$

where k is a nonzero scalar, S and S' are 1 for finite points in the plane, and 0 for infinite points. Since homogeneous coordinates are equivalent up to scalar multiples, the transformation matrix can be multiplied by any nonzero constant and still represent the same mapping, and therefore has only 8 independent parameters. A nonsingular projective mapping that is linear in homogeneous coordinates is called a *homography.* Matrices representing homographies form a group under matrix multiplication and matrix inverse.

Because they are linear, invertible and closed under composition, homographies greatly simplify the analysis of projective mappings. Figure 1 shows a familiar computer vision scenario where pictures are taken of a single planar surface by two cameras from different viewpoints.

Under the pinhole camera model, each camera effects a homography (in fact a *perspectivity*) from the object plane to a pinhole image plane, with all point correspondences lying on lines passing through the focal point.[1] The perspectivities are labeled $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$

[1]In this paper the term *point correspondence* denotes two points related by any plane to plane homography, not just image to image mappings.

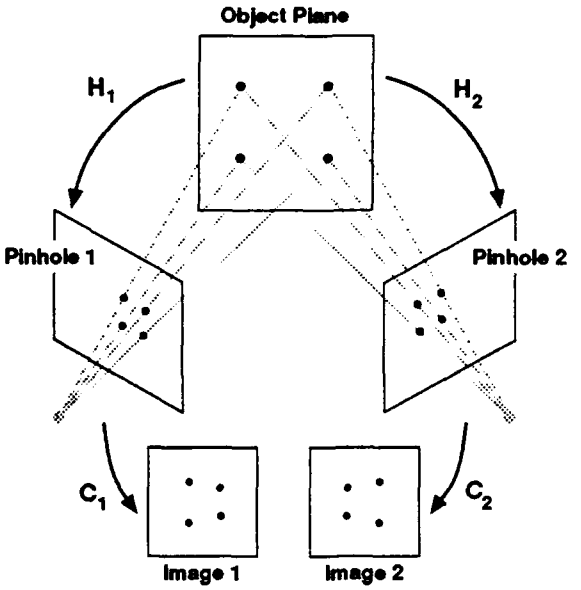


Figure 1: An object plane viewed by two cameras whose deviations from the pinhole camera model can be described by linear camera parameters. Corresponding points in any two planes in this diagram are related by a homography. Refer to the text.

in Figure 1. In actuality, the pinhole camera model does not adequately characterize images produced by real cameras. If the deviation of a particular camera from the pinhole model is governed by linear camera calibration parameters, the resulting image is an affine transformation of the pure pinhole projection [Horn86]. These transformations are labeled $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ in the figure. Affine transformations are a subgroup of the homography group, and the composition of a pinhole perspectivity followed by an affine deformation yields yet another homography. It is therefore easy to derive the transformation between any two planes in the diagram; for instance the transformation mapping points in image 1 into corresponding points in image 2 is $\boldsymbol{C}_2\boldsymbol{H}_2\boldsymbol{H}_1^{-1}\boldsymbol{C}_1^{-1}$.

Early research in the field studied the homography $\boldsymbol{H}_1\boldsymbol{H}_2^{-1}$ relating pinhole images of coplanar points, and showed that a decomposition of the homography matrix allows recovery of the relative positions of the two cameras with respect to each other and to the object plane [Faug88, Tsai82]. Recent work has focused on the object plane to image plane mapping $\boldsymbol{C}_i\boldsymbol{H}_i$, and more importantly its inverse $\boldsymbol{H}_i^{-1}\boldsymbol{C}_i^{-1}$ which backprojects image plane points to their appropriate object plane positions regardless of camera location or linear distortion parameters. This backprojection lies at the core of recent work by Mohr [Mohr91, Mohr90], as well as the approach to model extension in Section 3.

Unfortunately, the above analysis can not be carried out for cameras dominated by nonlinear lens distortion. Images of colinear points may then no longer be colinear, and the plane to plane mappings are no longer homographies. In the remainder of this paper nonlinear camera parameters are neglected; when such lens distortions are nonnegligable a preprocessing step must be performed to remove their effects [Gros90].

2.2 Estimation of a homography

The fundamental theorem of projective geometry states that a plane to plane homography is completely determined by the correspondences of 4 points, no three of which are colinear, or by 4 corresponding lines, no three of which meet at a point. In practice it is better to use as many point and line correspondences as possible to reduce errors in the estimated transformation caused by noise in the observed image data.

Faugeras and Lustman present a least squares approach to estimating a homography from at least 4 points [Faug88]. Each point to point correspondence adds two constraints on the eight independent parameters to be estimated. Using the notation of equation (2) these constraints are

$$\begin{aligned} aX + bY + c - X'(gX + hY + i) &= 0 \\ dX + eY + f - Y'(gX + hY + i) &= 0. \end{aligned}$$

Since possible solutions for the set of parameters are equivalent up to scalar multiples, a further constraint like $i = 1$ is imposed to provide a unique solution. One problem with the above constraints is that they are only valid when all the points are finite (S and S' in Equation 2 must not be zero). When one or both of the points in a correspondence are infinite, a modified pair of constraints can be used [Coll91]. Making sure that the constraint equations properly handle infinite points is important, since a standard coordinatization of the projective plane with respect to four basis points involves computing a homography mapping finite points to infinite points [Spri64]. Points at infinity also arise in practical applications (see Section 4).

Homographies preserve colinear structure, and the image of a line under a homographic mapping is the locus of its transformed points. The homogeneous coordinates representating a line are formed as the vector cross-product $\boldsymbol{p} \times \boldsymbol{q}$ of the homogeneous coordinates $\boldsymbol{p}$ and $\boldsymbol{q}$ of any two distinct points on the line. Under a general homograpy $\boldsymbol{A}$, the line passing through points $\boldsymbol{p}$ and $\boldsymbol{q}$ now passes through $\boldsymbol{Ap}$ and $\boldsymbol{Aq}$, and its homogeneous coordinates therefore become $\boldsymbol{Ap} \times \boldsymbol{Aq}$. It is straightforward, though slightly tedious, to show that for any nonsingular 3×3 matrix $\boldsymbol{A}$,

$$\boldsymbol{Ap} \times \boldsymbol{Aq} = [\boldsymbol{A}^{-1}]^T(\boldsymbol{p} \times \boldsymbol{q}).$$

This means that the homography matrix that maps line coordinates to line coordinates under a given projective mapping is the transpose of the inverse of the matrix mapping points to points, and vice versa.

Using the above relationship a line homography can always be converted to the corresponding point homography. Since line segment extraction is a more global process than point extraction, it seems reasonable to expect a point homography derived by transforming an estimated line homography to be more accurate than one estimated directly from points.

3 Planar Surface Model Extension

In this section an approach to planar model extension is described that assumes at least four linear features (points or lines) on the object plane are already known. The transformation mapping known object features to their corresponding images is a homography containing information about the camera location and imaging parameters. By inverting this transformation the positions of new points and lines on the object face are determined with respect to the known reference features, without having to solve first for camera location or intrinsic calibration parameters.

This approach is similar to one used by Mohr [Mohr90]. Mohr locates points on an object surface using pairs of cross ratios between an object point and four known object locations. Since the cross-ratio is invariant under homographies, the values in a cross-ratio pair can be computed directly from the image. It is possible to rewrite the mapping effected by Mohr's cross-ratio algorithm as a homography matrix. When exactly four point correspondences are used, the homography estimated using the least squares approach of Section 2.2 reduces to that used by Mohr. When more than four point or line correspondences are known, direct least squares estimation of the homography matrix is probably more accurate. Furthermore, the new data fusion approach presented in Section 4 allows positions estimated from several images to be merged to derive ever more accurate point and line positions from noisy observations.

An image sequence from the pose estimation literature was chosen to illustrate model extension [Kuma90]. Figure 2 shows a typical image from a sequence of 20 images taken by mounting the camera on a PUMA robot arm and rotating the arm 4 degrees between consecutive views. Ground truth data for the labeled points was measured in feet to two decimal places ($0.01\text{ft} \sim 1/8\text{in}$). This sequence has been analyzed by Kumar using a robust pose estimation algorithm, using ground truth positions of the 12 points marked with crosses to estimate camera pose, then estimating depths for the 20 points marked with circles.

The reference points used by Kumar occur in clusters of 4 points on 3 different planar surfaces. In the following experiment, each of these 3 surfaces was treated as a separate object plane containing 4 reference points (crosses) and 4 test points (circles). Cir-

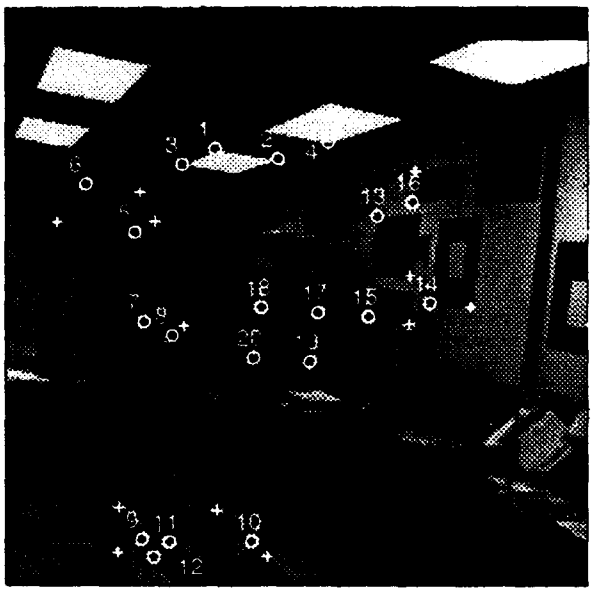


Figure 2: A typical image from Kumar's PUMA sequence.

cled points 1–4 and 17–20 were not used in this experiment. Since the 3D ground truth coordinate system was aligned with the natural axes of the room, a local 2D coordinate system for each object plane was formed by dropping the coordinate that didn't change for the points in that plane. In this way each measured 3D object point was converted to a 2D point in a local object plane coordinate system.

For each of the 3 object planes, the 4 reference points on that plane were matched by hand with their corresponding image points, and the linear system of equations in Section 2.2 was used to find the homography mapping image plane points onto the object plane points. The estimated homography was used to backproject the images of circled test points in the same plane back into the object plane, where their estimated positions were compared with the known ground truth locations. This process was repeated for each of the 20 images in the sequence.

Distance errors in points 9 through 12 are consistently low, around 1/3 of an inch, and never more than 3/4 inches.[2] This is because the chosen reference points are well spread out in that object plane, and completely surround the estimated test points. On the other hand, errors in points 13 and 15 are quite high: on average around 2 feet, and in some cases as much as 13 feet! This occurs because the reference points on that object plane do not adequately span the space of points to be estimated. In fact three of the reference points are very nearly colinear, a poor configuration for homography estimation.

To explore the relative accuracy of the estimated point positions, all distances between estimated points in a single object plane were compared to the ground truth distances, and percentage errors were computed. Figure 3 shows the results. The light grey curve for each plane shows the average percentage error taken over all point to point distances in that object plane for each image. Plane 2 (containing points 9–12) shows the most accurate results; the average percentage error over all images is 1.1%, around the level of noise in the ground truth measurements. Plane 1 (points 5–8) shows slightly worse errors, the average percentage error over all images is 3.5%. Finally, plane 3 (points 13–16) shows markedly bad errors, the average error over all images is 51%, but for some images the average error is as high as 200%.

Even in planes where points are estimated with high accuracy, the level of accuracy varies unpredictably from image to image. This is typical for a system that makes no use of previous estimates. In contrast, the smoother dark curve overlaid on each graph shows average percentage point to point distance errors when position estimates for previous images are combined with current position estimates using the data fusion technique described in the next section. A detailed analysis of the results appears in [Coll91].

4 Merging Geometric Information

A method for merging geometric information derived from multiple views is proposed in this section, based on fusing data points in the projective plane. From each image in the above sequence, a homography is estimated that backprojects image points onto the object plane, thereby providing an estimate for the location of each object point. Over multiple images, multiple location estimates are obtained. Each point location estimate in homogeneous coordinates represents a point in the projective plane; multiple location estimates for each object point form a sample of points in the projective plane, clustered around the point in the projective plane representing the homogeneous coordinates of the true object point location. This section describes a method for estimating the true point position from its sample cluster.

4.1 Probabilities in the Projective Plane

There are many ways to visualize the projective plane. In Section 2 the projective plane is described as the Euclidean plane augmented with a line of points at infinity. This is not the best way to visualize the projective plane, however, since the Euclidean plane is topologically open, while the projective plane is topologically closed. To see why adding points at infinity closes the

[2] All reported distance measurements lie entirely within the object plane; the notion of distance from the camera never arises in this framework.

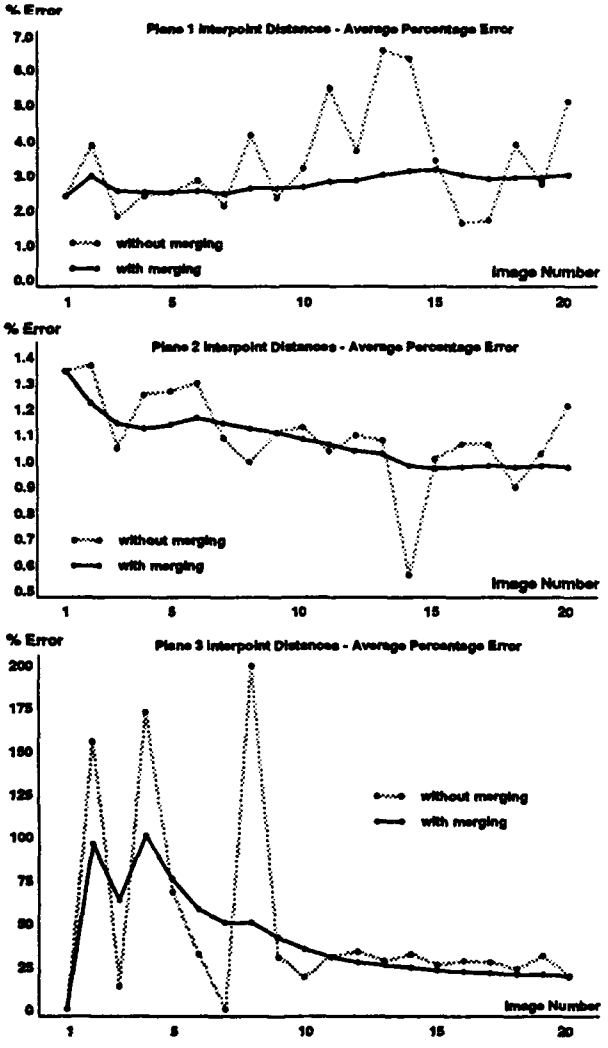


Figure 3: Graphs showing for each object plane the average percentage distance error between estimated interpoint distances and ground truth distances.

plane, consider a hypothetical traveler in the plane following a ray starting at the origin, going through point (x, y), and continuing out infinitely far. After finally "arriving" at infinity, the traveler is located at point $(x, y, 0)$ in homogeneous coordinates. But in homogenous coordinates this is the same point as $(-x, -y, 0)$, so the intrepid explorer can keep traveling "past infinity" in the same direction, eventually passing through point $(-x, -y)$ and finally returning to the origin.

Because of this wraparound effect, if the topology of the projective plane is ignored and it is treated as a Euclidean plane, a single cluster of points centered around a point at infinity will appear as two clusters infinitely far apart. Any estimation technique based on "averaging" these points using a Gaussian distribution in the plane will produce bad results in this case, because the unimodal Gaussian distribution is a terrible approximation to the underlying bimodal distribution.

Proper handling of points at infinity is not just of theoretical interest. Such points do arise in practice. For instance, parallel lines on an object plane often project in the image as lines that converge to a *vanishing point*. When backprojected from the image plane to the object plane, the vanishing point becomes a point at infinity, thus a cluster of vanishing point estimates becomes a cluster of points at infinity [Coll90a]. Furthermore, the homogeneous coordinates of lines in the image plane may have third coordinates that are zero or nearly so. Treated as points in the projective plane, these line coordinates are essentially points at infinity.

Since the projective plane is topologically closed, it is better to think of it as a closed 2D space like the surface of a sphere. More formally, define an equivalence relation $\sim$ on $R^3 - \{(0,0,0)\}$ such that $(x_1, x_2, x_3) \sim (y_1, y_2, y_3)$ iff there exists some nonzero k such that $x_i = ky_i, i = 1, 2, 3$. The projective plane P^2 is defined as the the quotient space[3]

$$(R^3 - \{(0,0,0)\}) \,/\, \sim .$$

Viewing R^3 geometrically as Euclidean 3-space, each member of the quotient space is an equivalence class of points along an infinite line through the origin (excluding the origin itself which would otherwise need to be a member of all the equivalence classes). Consider now the surface of the unit sphere $S^2 = \{(x_1, x_2, x_3) | x_1^2 + x_2^2 + x_3^2 = 1\}$, and form the quotient space $S^2 \,/\, \sim$. Each equivalence class now contains one pair of diametrically opposite points. Equating these equivalence classes with those of the projective plane in the obvious way shows that the surface of the unit sphere with antipodal points equated is isomorphic to the projective plane.

The most important benefit to come from this isomorphism is that it allows probability distributions on the sphere to be reinterpreted as distributions in the projective plane. Since diametrically opposite points on the sphere must be treated as equivalent in order to represent the projective plane, an appropriate distribution must possess the property of antipodal symmetry, i.e. the probability value at any point on the sphere must be the same as the probability at the diametrically opposite point.

A useful characterization of distributions on the sphere is presented in Beran [Bera79]. Beran considers exponential distributions on the sphere, that is, distributions of the form $exp\{P\}$ where P is a polynomial evaluated over the surface of the sphere. This assumption is not as restrictive as it seems, since any strictly positive function F on the sphere can be represented as $exp\{ln\{F\}\}$. One good reason for considering ex-

[3]Note this is also the space of homogeneous coordinates.

ponential distributions is their ease of use in maximum likelihood estimation [Mend87].

Assuming a distribution of the form $exp\{P\}$, Beran notes that the polynomial P can be decomposed using spherical harmonics, analogous to the way polynomials in Euclidean space are decomposed using Fourier analysis. If the distribution is required to have antipodal symmetry, all odd order harmonics are identically zero. This leaves an expression $exp\{Y_0+Y_2+Y_4+\ldots\}$. The zeroth harmonic is a constant, so the $exp\{Y_0\}$ term can be factored out and absorbed into the distribution's normalization constant. Therefore, the low order approximation to any antipodally symmetric exponential distribution on the sphere is of the form $exp\{Y_2\}$. A distribution having this form has already been studied in the statistical literature, where it is called Bingham's distribution [Bing74].

Bingham's distribution can be described as a trivariate Gaussian vector with zero mean and arbitrary covariance matrix, conditioned on the length of the vector being unity. Bingham's distribution thus represents the portion of a trivariate Gaussian distribution that intersects the surface of the unit sphere, with varying ellipsoidal shapes of the underlying Gaussian contours producing a variety of distributional forms on the sphere (see Figure 4). Bingham's distribution has been used previously in a computer vision setting to represent uncertainties in line and plane orientations estimated from vanishing point analysis and stereo line correspondences [Coll90b, Coll90a].

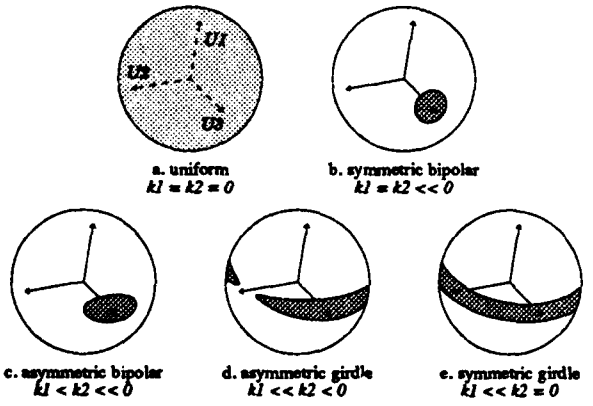


Figure 4: Bingham's Distribution – representative contours for varying shape parameter magnitudes.

4.2 Data Fusion in the Projective Plane

A point or line location estimate in homogeneous coordinates represents a point in the projective plane; multiple estimates form a sample of data points in the projective plane. To fuse data points in the projective plane each point is assumed to be a noisy observation of the true point location. The analysis of the last section shows that Bingham's distribution is a first order approximation to any noise process in the projective plane; therefore it is assumed that observed points are corrupted by a Bingham noise process, centered about the true point location.

Normalizing the homogeneous coordinates of each sample point yields an antipodal pair of points (or equivalently, an axis) on the unit sphere. Assuming a manageable level of noise, the normalized sample points form a cluster of points on the sphere distributed according to a bipolar Bingham distribution (Figures 4b and 4c). Assuming no bias in the observations, the polar axis of this distribution should coincide with the normalized homogeneous coordinates of the true point in the projective plane. An estimate of the true point position can therefore be obtained as an estimate of the polar axis of the Bingham distribution that best fits the normalized sample points. The most common method for estimating a distribution's parameters from a sample of observations is maximum likelihood estimation.

Relevant statistics for Bingham's distribution can be summarized as follows; a more detailed presentation can be found in [Coll90b, Bing74]. Given a set of n unit vectors $\phi_i = \langle x_i, y_i, z_i \rangle$ assumed to be distributed according to Bingham's distribution, a sufficient statistic for the orientation and shape parameters of the distribution is the sample second moment or *scatter* matrix

$$M = \frac{1}{n}\begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i z_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i z_i \\ \sum x_i z_i & \sum y_i z_i & \sum z_i^2 \end{bmatrix}.$$

Since the scatter matrix is a symmetric real matrix, it can be decomposed into $M = A\Lambda A^t$, where $A = [a_1, a_2, a_3]$ is an orthogonal matrix of eigenvectors, and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3)$ is a diagonal matrix of corresponding eigenvalues with $\lambda_1 \leq \lambda_2 \leq \lambda_3$ summing up to 1.

It can be shown that the maximum likelihood estimate of the pole of a bipolar Bingham distribution is the eigenvector associated with the largest eigenvalue, or vector a_3 using the above convention. Equations for computing confidence regions on the sphere can be found in [Coll90b, Bing74].

5 Comments and Future Work

An approach to planar model extension has been presented that uses apriori knowledge of the relative positions of at least four coplanar points or lines to derive the positions of other points and lines on the same plane in a way that is invariant to relative camera location and intrinsic camera parameters. One of the main contributions of this work is the development of an appropriate methodology for fusing geometric information in the projective plane.

The approach to data fusion described in Section 4 uses maximum likelihood estimation to fit a set of distribution parameters to a point sample. This method implicitly assumes that all points in the sample are independent, and identically distributed. While the independence assumption may be a necessary evil, points in the sample will probably *not* be identically distributed, since some extracted image features are more accurate than others. Future work will address ways of combining point estimates of different, but estimable, accuracy.

As with other projective geometric algorithms for estimating structure based on homographic transformations, finding correct point and line correspondences is a crucial problem. Faugeras and Lustman present a hypothesize and test paradigm [Faug88]. Initial guesses are made of sets of geometric primitives that may be coplanar, and an interframe transformation matrix is estimated from them. To the extent that other features in the image are transformed in a consistent way, they are added to the hypothesis set and a revised transformation is computed; if no matches consistent with the transformation estimate are found, the current hypothesis is dropped. This strategy could be adapted to the present task.

The model extension work described here requires apriori knowledge of at least four reference points or lines in each plane in the scene, and only points and lines on those planes can be reconstructed. An important item for future research is extensions for estimating relative locations of points outside the planes of reference. Mohr [Mohr91, Mohr90] shows that under limited noise complete 3D reconstructions are possible using reference points on two different object planes. This seems a promising line of research.

References

[Bera79] R. Beran, "Exponential Models for Directional Data," *The Annals of Statistics*, Vol. 7, No. 6, 1979, pp. 1162–1178.

[Bing74] C. Bingham, "An Antipodally Symmetric Distribution on the Sphere," *The Annals of Statistics*, Vol. 2, 1974, pp. 1201–1225.

[Coll91] R.T. Collins, "Model Acquisition and Extension in the Projective Plane," COINS Technical Report, Computer and Information Sciences, University of Massachusetts.

[Coll90a] R.T. Collins and R.S. Weiss, "Vanishing Point Calculation as a Statistical Inference on the Unit Sphere," *Proc. Third International Conference on Computer Vision*, Osaka, Japan, December 1990, pp. 400–403.

[Coll90b] R.T. Collins and R.S. Weiss, "Deriving Line and Surface Orientation by Statistical Methods," *Proc. Darpa I.U. Workshop*, Pittsburgh, PA., Sept. 1990, pp. 433–438.

[Faug88] O.D. Faugeras and F. Lustman, "Motion and Structure from Motion in a Piecewise Planar Environment," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 2, 1988, pp. 485–508.

[Gros90] W.I. Grosky and L.A. Tamburino, "A Unified Approach to the Linear Camera Calibration Problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, 1990, pp. 663–671.

[Horn86] B.K.P. Horn, *Robot Vision*, MIT Press, Cambridge, MA., 1986.

[Kuma90] R. Kumar and A.R. Hanson, "Pose Refinement: Application to Model Extension and Sensitivity to Camera Parameters," *Proceedings Darpa I.U. Workshop*, Pittsburgh, PA., September 1990, pp. 660–669.

[Mend87] J.M. Mendel, *Lessons in Digital Estimation Theory*, Prentice-Hall Signal Processing Series, Prentice-Hall, Inc., NJ. 1987.

[Mohr90] R. Mohr and E. Arbogast, "It Can be Done Without Camera Calibration," *Pattern Recog. Letters*, V. 12, 1990, pp. 39–43.

[Mohr91] R. Mohr and L. Morin, "Relative Positioning from Geometric Invariants," *Computer Vision and Pattern Recognition*, Maui, Hawaii, June 1991, pp. 139–144.

[Spri64] C.E. Springer, *Geometry and Analysis of Projective Spaces*, W.H. Freeman and Company, San Francisco, 1964.

[Tsai82] R.Y. Tsai, T.S. Huang, and W. Zhu, "Estimating Three-Dimensional Motion Parameters of a Rigid Planar Patch, II: Singular Value Decomposition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 30, No. 4, 1982, pp. 525–534.

Subsection G:

Active Vision and Visual Strategies

Where to Look Next using a Bayes Net: An Overview

Raymond D. Rimey*
The University of Rochester
Computer Science Department
Rochester, New York 14627

1 Introduction

Our goal is to formalize and implement mechanisms for true task-oriented (not task-specific) vision. In our model of a task-oriented vision system, a question is first asked about the scene. The system determines what scene information would be sufficient to answer the question. The level of detail sufficient to answer the question may vary among the different pieces of information extracted from the scene. Specific vision modules are sequentially brought to bear on selective areas of the image, the selection and exact processing of each vision module depending on the results of previously executed vision modules. Each module produces a partial representation of the (minimal) information needed to answer the question, and these results are combined to produce the answer. Table 1 summarizes the key differences between task-oriented vision and the classical, passive approach to computer vision.

A "where to look next" capability can act as the cognitive executive for active vision. If an active computational agent is subject to an information load that can overwhelm its resources, the executive can allow it to ignore irrelevant stimuli, choose its tasks wisely, survive, and achieve its goals. Alternatively, "where to look next" can simply save time and effort in doing visual jobs that in humans require attentional shifts, such as radiograph and CAT scan interpretation, photo interpretation, traffic monitoring, etc. Last, the task-oriented approach should make for more dependable vision performance built from more general (less domain-specific) vision tools. The claim is that current vision modules, even relatively simple ones, can become useful and robust when they are carefully applied in a specific context.

Our approach has as background a large amount of research into visual attention, classical work in eye movements, and recent advances in active vision, including camera movements and foveal - peripheral sensors. Specifically, our tools are decision theory, utility theory, and Bayesian probabilistic models [3, 4, 6, 7]. Two recent key developments are Bayes nets [9], and influence diagrams [9, 11]. Applications using these new techniques are beginning to appear. The first large experimental system that applied Bayes nets to computer vision is by Levitt [8]. The formulation of that system using influence diagram techniques is discussed in [2]. A sensor/control problem involving a real milling machine is solved using influence diagram techniques in [1]. A special kind of influence diagram, called a temporal belief network, is discussed in [5], and is being studied for an application in sensor based mobile robot control.

In what follows we present the basic framework of a task-oriented computer vision system, called TEA, that uses Bayes nets and a maximum expected utility decision rule. Knowledge about the scene and about the nature of the specific task given to the system are represented in the Bayes net. The decision of what vision modules to run is made using a value/cost utility measure, where value is based on mutual information measured between nodes in the Bayes net that correspond to actions and to the goal of the task. We introduce a new method for incorporating relational knowledge both into the Bayes net and into the utility measure. The decision of what areas of the scene to run a vision module on can be made using this relational knowledge. TEA models camera movements and distinguishes between vision modules that operate either on foveal or on peripheral image data.

Experimental results are presented from the TEA-0 system, our initial implementation of the general TEA framework. We also outline TEA-1, which uses a richer knowledge representation to support more complex visual tasks. The TEA systems solve the "where to look next" problem, enabling us to study a problem we call "how to look", and by combining our solutions to these problems we expect to build a true task-oriented vision

Passive vision	Task-oriented vision
use all vision modules	use only some vision modules
process entire image	process areas of the image
maximal detail	sufficient detail
extract representation first	ask question first
answer question from representation data	answer question from scene data
unlimited resources	resource limitations

Table 1: Key differences between passive vision and task-oriented vision.

*This material is based on work supported by the National Science Foundation under Grants numbered IRI-8920771 and IRI-8903582. The Government has certain rights in this material.



Figure 1: An example scene in the application domain.

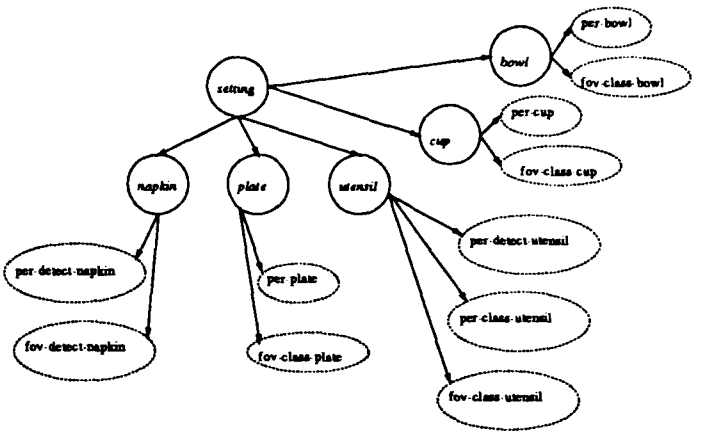


Figure 2: An example of a Bayes net that describes a place setting. Action nodes are drawn using dotted lines.

system. This paper is intended to be mainly an overview of our work. A fuller treatment appears in [10].

2 Preliminaries

The TEA system runs by iteratively selecting the evidence gathering action that maximizes an expected utility criterion:

1. List all the executable actions.
2. Select the action with highest utility.
3. Execute that action.
4. Attach the resulting evidence to the Bayes net and propagate its influence.
5. Repeat, until the task is solved.

The following section expands on aspects of the above algorithm. But first we present the application domain that we are currently using.

The approach is applicable whenever the scenes obey regularities or have structure that can be captured in the semantic data structures supporting Bayesian inference. One example is biomedical images that reflect known relationships and properties of anatomy. Another is aerial views of certain cultural areas such as industrial sites or airports. There is nothing inherently 2-D in the method. With the TEA system we currently use table settings. Figure 1 shows a typical table top scene.

We assume a spatially-varying sensor, which makes the "where to look next" question even more central. In the TEA system the peripheral image is a low-resolution image of the entire field of view from one camera angle, and the fovea is a small high-resolution image (*i.e.* window) that can be selectively moved within the field of view.

We assume the system can not view the entire scene at once. Often a camera movement must be made to an area of the scene that has not been been viewed before. The target location of such a camera movement must be determined via relations with other portions of the scene for which image data is (or previously has been) available. Following a camera movement the fovea is centered in the field of view, but afterwards the system can move the fovea within the field of view. The target location for a fovea movement is always within the field of view so it can be determined either from peripheral image data or by relations with other portions of the scene.

Our goal is to support many different visual tasks efficiently. Each task can be specified by asking a question about the scene: Where is the butter? Is this breakfast, lunch, dinner, or dessert? Is this an informal or fancy meal? How far has the eating progressed? Is this table messy? We are particularly interested in more qualitative tasks.

3 Framework for Solving Simple Tasks

3.1 Bayes Nets

A Bayes net is a way of representing the joint distribution of a set of variables in a way that is especially useful for knowledge representation (see [9] for details). For example, Figure 2 shows a highly simplified Bayes net that describes a place setting for a meal. Nodes in the net represent variables. Here, nodes drawn with solid lines denote parts of a place setting. The variable *setting* has four possible values ($breakfast, lunch, dinner, dessert$) that denote the respective types of meals. A *plate* can be either *paper* or *ceramic*. Links in the net represent conditional probabilities, for example the link from *setting* to *napkin* represents $P(napkin \mid setting)$, which says whether a napkin is expected at each of the possible meals.

The Bayes net formalism also includes a form of inference. Formally, belief in the values for node X is defined as $BEL(x) = P(x \mid \mathbf{e})$, where $\mathbf{e}$ is the combination of

all evidence present in the net. Elegant solutions have been developed [4, 9] for incorporating a single piece of evidence into the net and for propagating its effect to all other nodes in the net.

3.2 Adding Actions to a Bayes Net

A node X in our Bayes net has *action nodes* (drawn with dotted lines) connected to it, each representing a variable that is a visual action's "evidence report", which contains a score for each possible value of the Bayes net node X. An action's evidence report affects the BEL values of the parent node X. Before the action is executed, the action node is a "chance" node like most of the nodes in the net, and the node contains the BEL of the evidence report. After an action is (successfully) executed the action node is changed to be an "instantiated" node (see [9]) and set to the value of the evidence report. For example, in Figure 2 the `per-class-utensil` action might generate the following evidence report (4.9, 1.4, 3.2), which contains the scores for each of the objects, $(fork, knife, spoon)$. Action nodes are needed for computing the utility of actions given the current evidence present in the net.

All actions in the system are constructed from one or more low-level *vision modules*. Examples of some low-level vision modules are: color histogram matching and location-finding, (edge magnitude image) template matching, and a Hough transform for circles.

One or more vision modules may be used in a *visual action*. The Bayes net in Figure 2, used in one of our experiments with TEA-0, contains 11 visual actions. As an example, following is a more detailed summary of the actions related to plates:

- `per-plate`. Use Hough transform for plate-sized circles to detect plate in peripheral image. Detection succeeds or fails. The location is saved. Use color histogram to classify plate as paper (blue) or ceramic (green), using a window centered on the plate in the peripheral image.
- `fov-class-plate`. Plate location detected previously. Move fovea to plate. Use color histogram to classify plate as paper (blue) or ceramic (green), using fovea image data.

Complex actions like `per-plate` above are used in TEA-0, but are decomposed into simpler actions in TEA-1 (*e.g.* detect X, move peripheral window, classify X using peripheral window data, move fovea, classify X using foveal data, move camera). Some actions have *preconditions* that must be satisfied before they can be executed. TEA-0 uses only one kind of precondition: know the location of object X.

3.3 Adding Relations to a Bayes Net

Any node in a Bayes net that is bound to an object found in the scene will have the *location* of that object stored at that node. Otherwise, each node X has an *expected area* for the expected object associated with that node. An expected area is determined by applying geometric relations with each node Y_i connected to node X. A geometric relation between X and Y_i uses the location in node Y_i if it is available, otherwise the expected area at node Y_i is used. The expected area at node X is calculated strictly from relations; The location at node X is not used in the calculation of the expected area.

In TEA, an expected area is represented as a scalable bitmap denoting a subset of the planar scene area whose size depends on the object it is being related to. The resolution of the grid is currently the same as that of the peripheral images. TEA-0 allows relations between siblings, but in our latest work TEA-1 will have relations between parent and child nodes.

Each node Y_i produces an expected area for the object at node X. All these expected areas must be combined to obtain the final, single, expected area for the object at node X. In general it will be useful to characterize the relation depicted by the maps as "must-be", "must-not-be" and "could-be". Combination of two "must-be" maps would then be by intersection, and in general map combination would proceed by the obvious set-theoretic operations corresponding to the inclusive or exclusive semantics of the relation. In TEA-0 the relations are "could-be", and the maps are unioned.

3.4 Calculating an Action's Utility

Let the action node α have the node A as its parent, then the utility $U(\alpha)$ of an action α is of the form

$$U(\alpha) = \frac{V(\alpha)}{C(\alpha)}.$$

$C(\alpha)$ is the cost of executing the action:

$$C(\alpha) = r_A C_0(\alpha).$$

$C_0(\alpha)$ is the execution time of action α on the entire peripheral or foveal image, and r_A is the percentage of the image covered by the expected area of the object associated with node A. For foveal images, $r_A = 1.0$ since the entire fovea is always processed. Before any actions have been executed, no objects have been located, and so all r_A values are 1.0. Over time, as other objects in the scene are located and as more and tighter relations are established, the value of r_A will approach zero.

$V(\alpha)$ is meant to be the value of the action, how useful it is for achieving the task's goal. All actions in a computer vision system are *information gathering* actions. Therefore, the value of an action is strictly a measure of the information the action provides:

$$V(\alpha) = I(target, \alpha),$$

where the task's goal is represented by the node *target*. I is Shannon's measure of average mutual information (see, *e.g.* [9]):

$$I(X,Y) = \sum_x \sum_y BEL(x,y) log \frac{BEL(x,y)}{BEL(x)BEL(y)}$$

where

$$BEL(x,y) = BEL(x \mid y)BEL(y).$$

The values of $BEL(x)$ and $BEL(y)$ are respectively available at nodes X and Y in the Bayes net. The values of $BEL(x \mid y)$ can be calculated by temporarily instantiating node Y to each of its values, propagating beliefs, and taking the resulting $BEL(x)$ as $BEL(x \mid y)$ [9].

It is important to "look ahead" at the future impact of executing an action. Therefore we use the following "lookahead" utility function. Recall that action node α has the node A as its parent.

$$U(\alpha) = \frac{V(\alpha) + V(\beta)}{C(\alpha) + C(\beta)} + \sum_{X \in Rel(A)} \Delta U(X) \qquad (1)$$

where

$$\beta = argmax_{\gamma \in LocPre(A)} \frac{V(\gamma)}{C(\gamma)}$$

The first term in equation (1) accounts for the future value of establishing the location of an object. Action α might detect and locate an object, but not provide any information ($I = 0$) toward the task node in the Bayes net, however it does locate the object and thereby satisfy the preconditions of other actions that in turn will provide information useful for accomplishing the task. The interpretation of the first term in equation (1) is: Let β be the "best" action with a precondition that is satisfied by executing action α. The new utility of action α is an average over both α and β, more specifically an average of the value and cost of the two actions α and β. $LocPre(A)$ is the set of actions with the precondition of knowing the location of the object associated with node A.

The second term in equation (1) anticipates the impact of the expected areas that action α will generate (by establishing relations with other objects). $Rel(A)$ is the set of all nodes that are directly helped by location information about the object associated with node A. In other words, nodes A and X are siblings or a parent-child pair and they have a relation map defined between them. Each node in $Rel(A)$ contributes a term $\Delta U(X)$ to the utility:

$$\Delta U(X) = \max_{\gamma \in Actions(X)} (U(\gamma) * (1/r - 1)).$$

r is the percent reduction in the expected area for node X's object assuming that the location of node A's object is known, and is computed by applying the relation map, but it is applied using the expected rather than the known size of node A's object.

3.5 Experimental Example

The TEA-0 system is an initial implementation that follows the technical framework outlined above. TEA-0 works in a simplified domain (a single place setting) and solves the following task: decide which meal the place setting is for *breakfast*, *lunch*, *dinner* or *dessert*. The task is further simplified by assuming that the scene could contain only a napkin, plate, cup, bowl, and a single utensil. The entire scene can be viewed in one image so the system does not use camera movements. A relationship between the possible objects and the type of meal was contrived and encoded as the Bayes net model shown in Figure 2. The goal is to obtain high values for $BEL(setting)$. The scene was an overhead view of a single place setting, like one of those in Figure 1. The values of $BEL(setting)$ before any actions have been executed are:

$BEL(setting)$	
0.100	*bfast*
0.300	*lunch*
0.500	*dinner*
0.100	*dessert*

. The initial list of executable actions is:

$U(\alpha)$	
0.376946	`per-plate`
0.156381	`per-detect-napkin`
0.020900	`per-detect-utensil`
0.011891	`per-cup`
0.000509	`per-bowl`

The system ends up executing the following sequence of actions: `per-plate`, `per-detect-napkin`, `per-detect-utensil`, `per-class-utensil`, `per-cup`, `fov-class-utensil`, `per-bowl`, after which the task belief values correctly indicate that the place setting is most likely set for a dinner meal:

$BEL(setting)$	
0.004	*bfast*
0.066	*lunch*
0.867	*dinner*
0.063	*dessert*

. The remaining actions have little effect on the task belief.

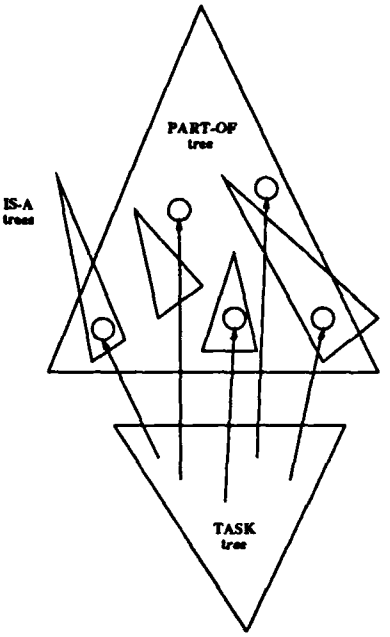


Figure 3: The organization of a large Bayes net used by TEA.

4 Framework for Solving More Complex Tasks

More complex questions will require a more complex Bayes net structure than used by TEA-0. The organization we are pursuing for the TEA-1 system is shown in Figure 3. It consists of three separate tree structures: a `PART-OF` tree, `IS-A` trees, and a *task* tree.

The `PART-OF` tree, an example of one is shown in Figure 4, models the physical structure of the scene. We assume the scene and all the objects in the scene can

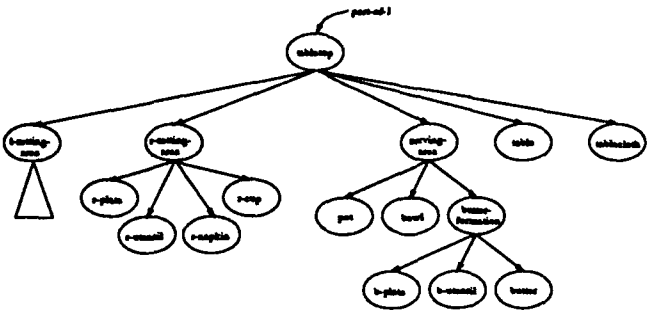


Figure 4: A PART-OF Bayes net.

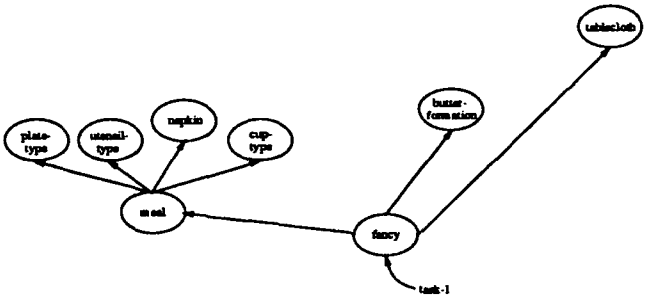


Figure 6: A *task* Bayes net.

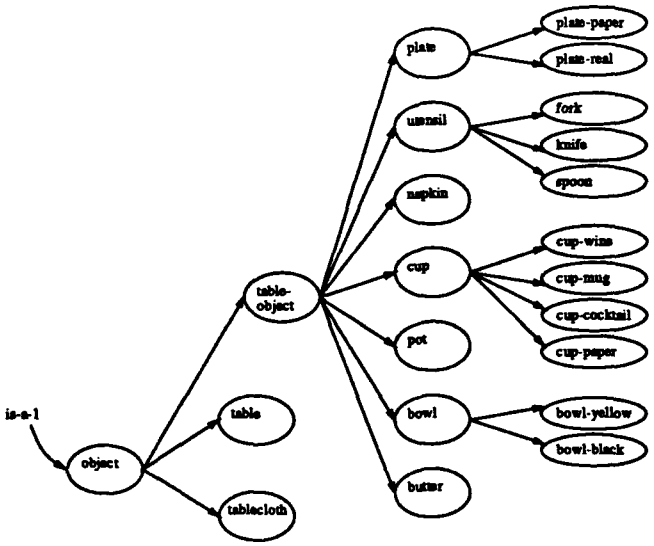


Figure 5: An IS-A Bayes net.

be modeled as a hierarchy of parts. All nodes in this net have the same set of possible values: *present* and *notPresent*. The conditional probability on each network link indicates the likelihood that a subpart exists. Each object's location and expected area are stored within the object's node in the `PART-OF` network.

An `IS-A` tree (for example, Figure 5) models an abstraction hierarchy for each instance of an object in the scene. The `IS-A` net is a special kind of network because the leaf nodes are mutually exclusive (*i.e.* the object can only be one thing). Belief nets with this special property have been developed [4, 9].

One of our scientific goals is to make a tight formal and practical coupling between "task specific knowledge" and visual actions. Task specific knowledge is contained in the *task* net (for example, Figure 6), and is thus distinguished from other types of knowledge. One feature of task knowledge is that subtask nodes could be shared by several tasks. Questions such as "Is this a fancy meal?" may be answered using a range of image clues. Some simple tasks, such as "Where is the butter?", do not require a task tree since they only involve one particular node in a tree.

We want to use the task tree to add task-specificity to the utility function: basically the idea is to relativize the utility calculation to the task by "projecting" the knowledge in the PART-OF and IS-A trees onto the task tree and computing utilities there. We are in the early stages of developing the notation, semantics, and implementation of the interacting trees. We wish to develop general formulae for these utilities, and to study the benefits gained by more complex utility functions, as opposed to simpler utility functions and more complete planning.

5 Conclusions and Future Work

We are pursuing two main streams of work. One stream develops the TEA systems, a progression of systems that support increasingly sophisticated task-oriented vision by providing solutions to the "where to look next" problem. The second stream of work uses and extends the TEA framework to explore broader and more advanced issues in task-oriented vision: foveal - peripheral vision algorithms, qualitative visual tasks, limited-context vision algorithms that gain in robustness or accuracy by being applied in well-understood circumstances, incremental visual actions whose results monotonically improve as more time is spent on them, representations of 3-D and dynamic spatial relations, head-shifting and viewpoint planning, and processor scheduling.

5.1 "Where to Look Next"

TEA-0: Using relations. An initial version of TEA-0 has been implemented and it should give the basic idea of our approach. TEA-0 will be completed by implementing the full version of relations, and by enabling camera movements.

TEA-1: Projecting utilities through a task tree. The main feature of TEA-1 is the addition of multiple interacting trees, which permit the system to solve more complex tasks. A preliminary design for projecting utility calculations through the task tree is complete and we have begun implementing it. A deeper issue is that relational information should be used to modify probabilities and beliefs, not just costs. Relational evidence must then be formulated in a probabilistic framework, and expected areas used like beliefs.

TEA-2: Planning. TEA-0 and TEA-1 are "myopic", making decisions by only looking one step ahead. The anticipatory utility function is an improvement, trying to pack look-ahead into the utility of a single action.

Ultimately our problem involves full-scale planning, in which sequences of actions are evaluated as to their expected utility. We intend to develop a simple planning system (for computer vision) using Bayes nets. We do not propose "planning research" *per se*, but rather shall likely use some STRIPS-like planning algorithm. The idea is to substitute a search in action space rather than to try to pack all the intelligence into a (quasi-static) utility function.

5.2 "How to Look"

Limited-Context Vision Algorithms. One claim of this work is that vision algorithms can be more robust and reliable if they are known to work in a limited context. For example, TEA-0 can use simple color histograms for object identification only because it has foveated a small area of the image previously. Similarly restricting input to a small volume of space means geometric hashing can work more reliably. We want to explore limited context effects that arise naturally in task-oriented vision when the vision problem is known to be simplified or better specified than normal (by camera actions, foveal processing, and generally by satisfaction of preconditions).

Incremental actions. We want to investigate vision modules that can run for different periods of time, improving their results the longer they run (*e.g.* some scale space algorithms, multi-feature classifiers). Such vision actions are generalizations of TEA-0's peripheral - foveal actions which produce a peripheral result at one cost and follow it up with a foveal action for a further cost. An evidence/time function can quantify the incremental benefit of such an action. New control strategies should then emerge, such as running a set of incremental actions cyclically to attain the maximum evidence per unit time from the set.

5.3 Task-Oriented Vision

Our idea of a true task-oriented vision system will be achieved by bringing together solutions to the "where to look next" and the "how to look" problems.

Multiple Tasks. We plan to solve multiple tasks in any given domain using the same set of visual actions. This exercise will test the generality of our knowledge representations and visual actions and probably encourage us to extend and modify both. Also we expect to encounter interesting new problems for visual actions used in answering qualitative questions such as "Is this desk messy?".

Multiple Domains. We believe that a task-oriented vision system should be verified using more than one domain. We shall seek out other domains. A possible domain is model trains to be monitored on a more or less complex system of tracks. Another is monitoring or searching the laboratory space in 3-D and performing head movements as well as camera movements, and ultimately dynamic scenes. Medical images are another possibility emphasizing reliability as opposed to active vision. Expanding the domains will doubtless mean that visual actions need to be re-engineered and improved to apply more generally. Difficulties in encoding or coping with new domains will motivate extensions and modifications to our formalisms. New domains may necessitate the use of more complex knowledge representations, in particular non-tree Bayes nets.

Acknowledgements

Chris Brown, my advisor, has been invaluable to me throughout this work by providing crystal judgement and ideas and asking key questions. Peter von Kaenel worked on the Hough transform for circles and is currently building the modules that match models using straight lines and circular arcs. He also built several of the visual actions.

References

[1] A. M. Agogino and K. Ramamurthi. Real time influence diagrams for monitoring and controlling mechanical systems. In R. M. Oliver and J. Q. Smith, editors, *Influence Diagrams, Belief Nets and Decision Analysis*, pages 199–228. John Wiley and Sons, 1990.

[2] J. M. Agosta. The structure of Bayes networks for visual recognition. In *Uncertainty in AI*, pages 397–405. North-Holland, 1990.

[3] R. C. Bolles. Verification vision for programmable assembly. In *Proceedings: International Joint Conference on Artificial Intelligence*, pages 569–575, 1977.

[4] P. B. Chou and C. M. Brown. The theory and practice of Bayesian image labeling. *International Journal of Computer Vision*, 4(3):185–210, 1990.

[5] T. Dean, T. Camus, and J. Kirman. Sequential decision making for active perception. In *Proceedings: DARPA Image Understanding Workshop*, pages 889–894, 1990.

[6] J. Feldman and R. Sproull. Decision theory and artificial intelligence II: The hungry monkey. *Cognitive Science*, 1:158–192, 1977.

[7] T. Garvey. Perceptual strategies for purposive vision. Technical Report 117, SRI AI Center, 1976.

[8] T. Levitt, T. Binford, G. Ettinger, and P. Gelband. Probability-based control for computer vision. In *Proceedings: DARPA Image Understanding Workshop*, pages 355–369, 1989.

[9] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufman, 1988.

[10] R. D. Rimey and C. M. Brown. Where to look next using a Bayes net: An overview, incorporating geometric relations, and projecting utilities through a task tree. Technical Report 398, Department of Computer Science, University of Rochester, November 1991.

[11] R. D. Schachter. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, 1986.

Learning Knowledge-Directed Visual Strategies

Bruce A. Draper Allen R. Hanson Edward M. Riseman
Dept. of Computer Science
University of Massachusetts
Amherst, MA., USA. 01003*

Abstract

The goal of image understanding systems is typically the identification of objects in visual imagery and the establishment of the three-dimensional relationships among the objects and the viewer. Because of the variety and scope of knowledge pertinent to vision, however, the acquisition of both object models and interpretation strategies remains a major outstanding problem in model-based image understanding. We contend that learning techniques must be embedded in vision systems of the future in order to reduce or eliminate the knowledge engineering aspects of system construction, and present the Schema Learning System (SLS) as a prototype system for learning object recognition strategiess.

1 Introduction

The goal of image understanding systems is typically the identification of objects in visual imagery and the establishment of the three-dimensional relationships among the objects and the viewer. It is a generally accepted premise that, in many domains, the timely and appropriate use of relevant knowledge can substantially reduce the combinatorially explosive search encountered in establishing 'instance-of' relationships between image data and object classes in the knowledge base.

Because of the variety and scope of knowledge pertinent to vision, the acquisition of both object models and interpretation strategies remains a major outstanding problem in model-based image understanding. While many vision algorithms at the low and intermediate levels are available, successful use of knowledge in image understanding applied to outdoor scenes requires a careful hand-crafting of the knowledge base ([5]). Typically this requires specifying, for each object class, both a description of the generic object and one or more recognition (control) strategies for instantiating instances of the object to image data.

The success of many knowledge-based image understanding systems can be traced to a "small world" assumption, in which the number of objects in the domain are few, the constraints on their descriptions are tight, and a complete world model is at least a possibility. Consequently, special purpose systems are able to define, structure, and apply relevant task knowledge effectively. However, as the scope of a system broadens towards a domain-independent, general-purpose system, an unfortunate chain of events occurs: the size of the knowledge base increases, constraints on the object descriptions become looser to account for wider variability, the system must make fewer assumptions about the types of image descriptions necessary for matching, and the complexity of matching increases substantially.

There are really two issues being discussed here: the structure of object and control knowledge in vision systems, and the acquisition of this knowledge. The next section briefly describes the knowledge component of the VISIONS image understanding system ([8, 9]) known as the Schema System ([5]), from the point of view of knowledge structuring and control. Subsequent sections discuss the role of learning in the automatic acquisition of portions of the knowledge base. It is our contention that learning techniques must be embedded in vision systems of the future in order to reduce the cost of knowledge base construction.

2 The VISIONS Schema System

The success of systems based on the "small world" assumption has led us to adopt a design philosophy that partitions both knowledge and computation at a coarse-grained semantic level. In the VISIONS system, the coarse-grained knowledge is encapsulated in schemas, where each schema is specialized to a single object class. This encapsulation permits schemas to be "experts" in the recognition of instances of an object class and permits the wide range of control strategies necessary for objects to be represented in a natural way.

Schema instances are invoked for each object class hypothesized to be in the image data. These instances execute independent (potentially concurrent) processes called recognition strategies and communicate asyn-

*This work was supported by DARPA under TACOM contract DAAE07-91-C-R035, RADC under contract F30602-91-C-0037 and the NSF under grant CDA-8922572.

chronously through a global blackboard. The control component of each schema directs the application of general purpose procedures, called knowledge sources, to gather the "right kind" of support for (or against) its hypothesis. Competition and cooperation among the schema instances results in the combination of multiple, independent "object experts" into a large scale system which constructs internally consistent interpretations.

2.1 Schema System Components

The schema system consists of five basic components: the schema hierarchy, the blackboard, the knowledge sources, the interpretation (control) strategies, and mechanisms for evidence representation and combination. Each of these is discussed very briefly in the following sections; more detail may be found in [5].

2.1.1 The Schema Hierarchy

The schema system partitions both knowledge and computation in terms of natural object classes for a given domain. Schemas reside in class and part/subpart hierarchies; each class of objects and object parts has a corresponding schema which stores all object and control knowledge specific to that class. Knowledge about expected object contexts and relationships to other objects is represented in the system by extending the concept of an object to include contextual or scene configurations; as objects, these entities also have schemas. A subcontext or "sub-scene" is like an object part; it is related to its parent scene or context in predicatable ways.

2.1.2 Knowledge Sources

Knowledge sources are general-purpose procedures that generate the levels of abstract image descriptions required for image understanding. Knowledge sources span the gamut of traditional techniques in image processing (e.g. region, line, curve, and surface extraction, feature measurement, etc), through intermediate level processes such as initial object hypothesis generation and grouping operations to generally useful tools and techniques such as graph matching. The compile-time arguments and parameters supplied to general-purpose knowledge sources as part of the recognition strategy may specialize them for particular purposes.

2.1.3 Interpretation Strategies

Interpretation strategies, or simply strategies, are control programs that run within each schema. Strategies procedurally encode knowledge about which knowledge sources to apply and in what order to apply them. In order to make maximal use of parallelism, schemas may have multiple concurrent strategies. These strategies may correspond to different methods for recognizing an object or to different conditions under which recognition must take place. Schemas can also contain strategies for different subtasks, such as initial hypothesis generation and hypothesis verification, as well as for managing the internal bookkeeping details of the schema, such as updating the global blackboard when necessary and detecting and resolving conflicts related to the hypothesis.

Each schema instance acquires information pertinent to the hypothesis it is pursuing. Some of this information is generic, to the extent that its semantics are not object dependent. For example, the degree of confidence in a hypothesis, as well as its (2D) image location and (3D) world location, is generic information. Every object hypothesis has a confidence level and an image location, and most have a meaningful 3D location. The generic information about an object hypothesis is recorded in a global hypothesis.

Most of the information acquired by a schema instance, on the other hand, is object specific. Information about how well an image region matches an expected color, for example, is non-generic since its importance depends on the object model. A color match may be important for finding trees, but less so for recognizing automobiles. For this reason, all of the information about which KSs support a particular hypothesis and which do not is considered private to the schema instance, and is not included in the global hypothesis.

2.1.4 Blackboard Communication

The schema system is built around a global blackboard. The global hypotheses written to the blackboard represent the image interpretation as it evolves. Schemas communicate with each other by writing to and reading from the blackboard, dynamically exchanging information about their respective hypotheses. Although the blackboard is divided into sections corresponding to the object classes (rather than processing levels, as in other systems), schemas may read and write freely over the entire blackboard. The division into sections gives some assurance that a schema will not have to search through a large number of irrelevant messages. At the same time, each schema instance maintains its own local blackboard for recording private information.

The distinction between the global and local blackboards was motivated both by computational and knowledge engineering concerns. Computationally, most of the information generated by an interpretation strategy concerns which KSs have been run, what each KS returned, etc. While this information is crucially important within the schema instance for making dynamic control decisions, it is of little importance to other schema instances. If the strategies associated with multiple concurrent schema instances continually dump this information to the global blackboard and then read it back again, the blackboard quickly becomes a computational bottleneck. The local blackboards alleviate this problem by reducing the message traffic on the global blackboard.

From a knowledge engineering viewpoint, the distinction between the global and local blackboards promotes modularity. By allowing only the strict "global hypothesis" protocol to be exchanged between schemas,

the schema system encourages modularity. Each schema can maintain local information in an idiosyncratic manner on its local blackboard, allowing the schema designer the freedom of any appropriate knowledge representation and control style. At the same time, because schemas communicate with each other only through global hypotheses, the designer of a new schema is assured of a smooth join to the remainder of the system.

2.1.5 Evidence Accumulation

The current version of the schema system takes a particularly simple view of evidence representation and combination. Confidence values lie along a coarse, five point ordinal scale: 'no evidence', 'slim-evidence', 'partial-support', 'belief', and 'strong-belief'. When combining evidence, a heuristic mechanism is used that involves the specification of key pieces of evidence that are required to post an object hypothesis with a given confidence to the global blackboard. Subsets of secondary evidence are used to raise or lower these confidences. Specifications of these subsets, and the effect their confidence has on the overall confidence, is part of the knowledge engineering effort involved in constructing a schema. Although this method of evidence representation and accumulation may lack considerably from a theoretical point of view, it worked surprisingly well in interpretation experiments on images of New England house and road scenes [5].

2.2 Knowledge Engineering

Schemas are assembled by specifying (1) the appropriate set of knowledge sources to be used, (2) a set of strategies which conditionally sequence their application, and (3) a function to translate internal evidence into a confidence in the global hypothesis. One of the main impediments to wide scale experimentation with the schema system has been the time and energy required to design a schema. Schema construction can be viewed as an exercise in experimental engineering, in which prototype schemas are developed using existing system resources. These schemas must then be tested on a representative set of objects/images, failures noted and analyzed, and the schemas re-engineered to account for the failures. In many cases, the descriptive information provided by the knowledge sources may be inadequate. In this case, new knowledge sources must be developed and tested (often a major research effort in its own right), integrated into the system, and the schemas re-engineered to make use of the new information.

The problem of knowledge base construction has been a focus of research for several years. In artificial intelligence, researchers have focused on how to extract knowledge from experts; vision researchers have concentrated instead on how knowledge bases should be specified. By restricting the message types written to the global blackboard, the schema system enforces schema modularity in an attempt to make them easier to declare and improve. The SPAM project at CMU went even farther, developing a high-level language for describing objects ([12]). Work in Japan has involved both automatic programming efforts and higher-level languages for specifying image operations ([11]).

3 Vision and Learning

For the last two years we have taken a different approach to knowledge base development. Instead of making the knowledge base easier to program, we have decided to take the programmer out of the loop. Knowledge-directed vision systems should learn their own interpretation strategies.

As a first step toward achieving this goal the Schema Learning System (SLS; [6, 7]) has been designed. The task of is to learn interpretation strategies for the different object classes in a domain. In particular, the goal is to learn a strategy that minimizes the cost of object recognition, subject to accuracy constraints supplied by the user. For example, a user might request a strategy for recognizing the (3D) position and pose of a building, accurate to within 5%. SLS would respond by attempting to learn a recognition strategy that satisfies this goal and could be invoked whenever the user needs to locate a building.

As implied by the scenario above, SLS's operations can be divided into two parts: a compile-time (or "learning-time") component in which SLS develops recognition strategies, and a run-time component in which the interpretation strategies are applied to new images. In general, SLS has been designed to optimize run-time performance, at the expense of compile-time (learning) efficiency.

SLS's task is made easier by two simplifying assumptions. First, SLS learns to recognize instances of each object class independently. This is easier than learning concurrent, cooperating strategies. Second, SLS is given a set of parameterized knowledge sources from which to build its recognition strategies. Thus SLS is not required to learn new knowledge sources or 3D object models, but rather to learn control strategies for applying knowledge sources, including evidence weighting schemes. Implicit in this task statement is the possibility that the object model may not be completely accurate, and that some of the knowledge sources may be misleading or irrelevant.

3.1 Modeling the Interpretation Process

SLS, like the schema system before it, adopts the blackboard model of interpretation and views object recognition as a process of applying knowledge sources to hypotheses. Hypotheses are proposed statements about the image or its interpretation, whose type is determined by their *level of abstraction*. Common levels of abstraction for computer vision include: image, region, image line segment, and image segment group (all of which are 2D),

and world line segment, orientation vector, world segment group, surface patch, face, volume (all of which are 3D). An "interpretation" is a set of believed hypotheses at the level of abstraction requested by the user. Knowledge sources are image understanding procedures (e.g. region segmentation, line extraction or vanishing point analysis) that can be applied to one or more hypotheses.

SLS refines the blackboard model of interpretation by constraining knowledge sources to fall into one of two classes. *Generation knowledge sources* (GKSs) create new hypotheses at a given level of abstraction. For example, a stereo line-matching algorithm that produces a (3D) world line segment from a pair of (2D) image line segments is a GKS. *Verification knowledge sources* (VKSs) return discrete evidence values about hypotheses at the level of abstraction they apply to. An example of a VKS is a pattern matching algorithm that determines if the color or texture of an image region matches the expected color or texture of the object.

3.2 Recognition Graphs

Interpretation strategies are represented in SLS as generalized multi-level decision trees called *recognition graphs* that direct both hypothesis formation and hypothesis verification, as shown in Figure 1. The premise behind the formalism is that object recognition is a series of small verification tasks interleaved with representational transformations. Recognition begins with trying to verify hypotheses at a low level of abstraction, separating to the extent possible hypotheses that are reliable from those that are not. Verified hypotheses (or at least, hypotheses that have not been rejected) are then transformed to a higher level of abstraction, where a new verification process takes place. The cycle of verification followed by transformation continues until hypotheses are verified at the goal level of abstraction (as specified by the user), or until all hypotheses have been rejected.

The structure of the recognition graph reflects the verification/transformation cycle. Each level of the recognition graph is a decision tree that controls hypothesis verification at one level of abstraction by invoking VKSs to gather support for or against each hypothesis. When the decision tree determines that a hypothesis is reliable, a GKS transforms it to another level of abstraction, where the process repeats itself.

As defined in the field of operations research, decision trees are a form of state-space representation composed of alternating *choice states* and *chance states*. When searching for a path from the start state to a goal state, an agent is only allowed to choose where to go next from a choice state. If the current state is a chance state the next state is selected probabilistically[1]. The search process is therefore similar to using a game tree against

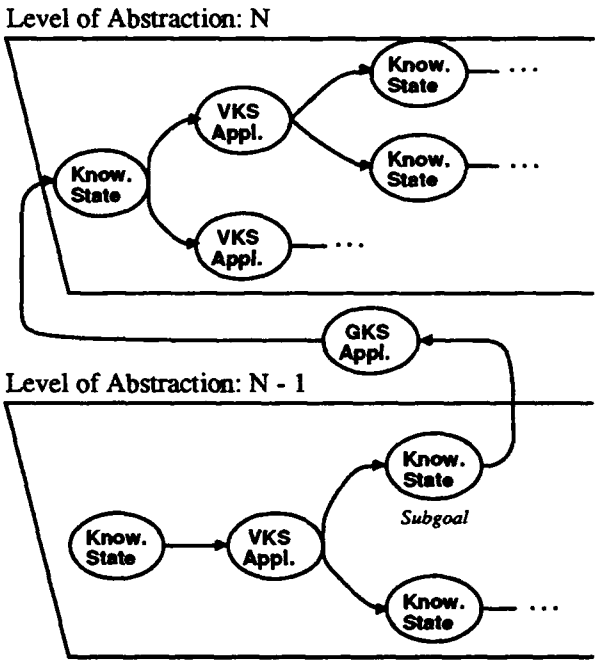


Figure 1: A recognition graph. Levels of the graph are decision trees that verify hypotheses using VKSs. Hypotheses that reach a subgoal are transformed to the next level of abstraction by a GKS.

a probabilistic opponent.

In SLS, the choice states are hypothesis knowledge states as represented by sets of hypothesis feature values. The choice to be made at each knowledge state is which VKS (if any) to execute next. Chance states in the tree represent VKS applications, where the chance is on which value the VKS will return. Hypothesis verification is an alternating cycle in which the control strategy selects which VKS to invoke next (i.e., which feature to compute), and the VKS probabilistically returns a feature value. Thus hypotheses advance from knowledge states to VKS application states and then on to new knowledge states. The cycle continues for each hypothesis until it reaches a subgoal state, indicating that it has been verified and should be transformed to a higher level of abstraction, or a failure state, indicating that the hypothesis is unreliable and should be rejected.

In general, SLS learns in advance what VKS to choose at each knowledge state in order to avoid making run-time control decisions. As a result, when SLS builds a recognition graph it leaves just one option at each choice node. Sometimes, however, the readiness of a VKS to be executed cannot be determined until run-time, in which case SLS will leave several options at a choice node, sorted in order of desirability[2]. At run-time the system will choose the highest-ranking VKS that is

[1] Operations research terminology is based on trees rather than spaces, so it refers to choice nodes and chance nodes rather than choice states and chance states, and to leaf nodes and root nodes rather than goal states and start states.

[2] This is just one of many complications that arise from multiple-argument knowledge sources. In general, we will

ready to be executed.

3.3 The Schema Learning System

The Schema Learning System (SLS) constructs interpretation strategies represented as recognition graphs. SLS is given (1) a set of parameterized knowledge sources; (2) a set of user-interpreted training images; and (3) a goal, in terms of a target representation and the required accuracy. It produces a recognition strategy that minimizes the expected cost of achieving the goal.

3.3.1 Exploration

SLS learns recognition strategies through a three-step process of exploration, learning from examples, and optimization. The first step, exploration, is algorithmically the least interesting. It exhaustively applies all available knowledge sources (both VKSs and GKSs) to the training images in order to estimate the expected cost of each knowledge source (measured as execution time) and the probability of each VKS result. The exhaustive exploration phase also produces as many correct interpretations as possible with the existing GKSs to serve as examples for the second phase of learning. At this point computational efficiency is unimportant, since the goal is run-time efficiency.

3.3.2 Learning from Examples

SLS's second step looks at the correct interpretations produced during exploration and infers from them a scheme for generating good hypotheses while minimizing the number of false hypotheses by tracing back the GKSs employed to produce each good hypotheses. For example, a correct 3D pose hypothesis might be generated by fitting a plane to a set of 3D line segments. If so, the pose hypothesis is dependent on the plane fitting GKS. It is also dependent on whatever GKS created the 3D line segments, and any GKSs needed to create its arguments, etc. The result of tracing back a hypothesis' dependencies is an AND/OR tree like the one shown in Figure 2. 'AND' nodes in the tree result from GKSs that require multiple arguments, such as stereo matching. 'OR' nodes in the tree occur when a hypothesis is redundantly generated by more than one GKS (or a single GKS applied to alternate hypotheses).

Each dependency tree is viewed as an example of how correct hypotheses are created. The example is generalized by replacing the hypotheses in the tree with their feature vectors. In other words, instead of viewing Figure 2 as showing how pose-10 was created during training, we interpret it as an example showing how poses can be created by a specific GKS (e.g. the GKS for fitting lines to planes) when applied to hypotheses with specific features (in this case, the feature values of 3D-lineset-1).

describe SLS as if all KSs took just one argument in order to keep the description brief; see [7] for a more complete description.

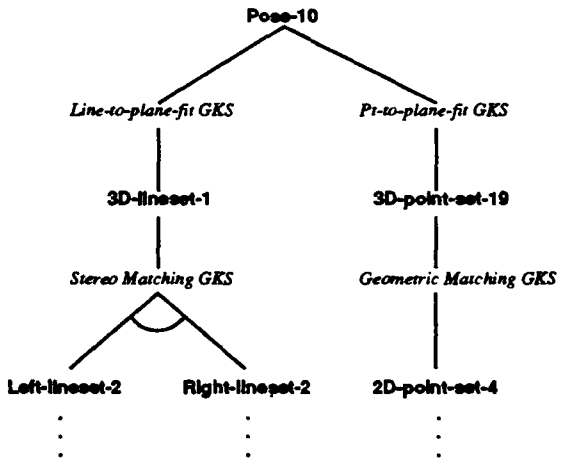


Figure 2: An example of a dependency tree showing the different ways that one correct pose hypothesis can be created during training.

Since the goal is to learn a strategy that will generate (and later verify) all instances of an object or object class, SLS collects the dependency trees of all the correct hypotheses into a single multi-sample tree by ANDing their root nodes together. By definition, any set of conditioned GKSs (i.e. GKSs with specific feature values as preconditions to the arguments derived from the training set) that satisfies this tree will generate all the correct hypotheses over the training images. However, there is no reason to believe that such a set of GKSs will generate only correct hypotheses; it will generate incorrect ones as well. Therefore, SLS's job in step two is to find a set of (conditioned) GKSs that satisfies the multi-sample dependency tree while minimizing the number of incorrect hypotheses generated.

SLS finds the optimal set of generation knowledge sources (GKSs) by converting the multi-sample dependency tree into disjunctive normal form (DNF) and selecting the conjunctive subterm that generates the fewest incorrect hypotheses. Because of the way the tree was constructed, the GKSs in any subterm are sufficient to generate correct goal-level hypotheses for every object instance in the training set.

The AND/OR dependency tree is converted into DNF by a standard algorithm that first converts its subtrees to DNF and then either merges the subterms (if the root is an OR node) or takes the symbolic cross-product of the subterms (if the root is an AND node). SLS, however, is designed to find just the minimal term of the resulting DNF expression; as a result, any time during the conversion process that a DNF has two subterms one of which is a logical superset of the other, the superset term can be pruned from the expression.

Readers may note that converting an arbitrary AND/OR tree to DNF is an exponentially expensive pro-

cess: in the worst case, a tree with N literals can produce a DNF expression with 2^N subterms. In the case of SLS, the pruning condition reduces the worst-case complexity to N choose $\lfloor \frac{N}{2} \rfloor$, but this is still exponential. The worst-case analysis, however, is largely inappropriate because it corresponds to random data. As long as the samples in the training set are visually similar (and as long as not all of the knowledge sources produce random results) the worst case will never arise.

3.3.3 Optimization

As was stated earlier, recognition graphs interleave verification and transformation, using VKSs to gather evidence to verify or reject hypotheses, and GKSs to transform them to higher levels of abstraction. By building a dependency tree from the training samples, converting it to DNF and picking the minimal subterm (measured by the number of incorrect hypotheses generated), SLS learned which GKSs to include in a strategy that generates correct hypotheses while minimizing the number of false alarms (and presumably cost). Just as important, it learned what evidence to require of a hypothesis before it should be transformed. The subterms of the DNF expression are GKSs constrained to be applied to hypotheses with specific sets of features, and these feature sets are the subgoals of the recognition process.

In the third step of the algorithm, SLS optimizes recognition by building decision trees for each level of abstraction that minimize the expected cost of reaching a subgoal or, conversely, of deciding that a hypothesis cannot satisfy any subgoal and should be rejected. This is achieved at each level by first laying out the graph of all possible sequences of knowledge states and VKS applications and then pruning it to leave just the tree that minimizes the expected cost.

For each level of abstraction, the initial graph layout begins with a start state. VKS applications are added for every VKS that can be applied to a hypothesis in the start state, and these VKS applications lead to new knowledge states, which in turn have more VKS applications attached to them, and so on. The expansion of the graph continues until it reaches either a subgoal knowledge state or a knowledge state that is incompatible with every remaining subgoal (i.e. a failure state).

Once the initial graph has been laid out, SLS begins to prune it by working backwards from the subgoal and failure nodes toward the start state. At each VKS application node it calculates the expected cost of reaching a subgoal or failure node from that particular application node. At each knowledge state, it finds which of the possible VKS applications has the lowest expected cost and removes the other VKSs from the list of candidates (in the event that the optimal VKS might not be executable at run time, it sorts the remaining VKSs in order of least to greatest expected cost rather than removing them.

More formally, we refer to the subgoal states and the failure states at one level of a recognition graph as the *terminal* states for that level. The cost of promoting a hypothesis from knowledge state n to a terminal state is called the Expected Decision Cost (EDC) of state n, and the expected cost of reaching a terminal state from state n using VKS k is the Expected Path Cost (EPC) of n and k. Since verification KSs return discrete values, we refer to the possible outcomes of a verification KS k as $R(k)$, and the probability of a particular value e being returned as $P(e|k,n), e \in R(k)$.

The EDC's of knowledge states can be calculated starting with the terminal states and working backwards through the recognition graph. Clearly, the EDC of a subgoal or failure state is zero:

$$EDC(n) = 0, \quad n \in \{terminal\ states\}.$$

The expected path cost of reaching a terminal state using a particular VKS is:

$$EPC(n,k) = C(k) + \sum_{e \in R(k)} (P(e|n,k) \times EDC(n \cup e))$$

where n is the knowledge state expressed as a set of feature values, $n \cup e$ is the knowledge state that results from VKS k returning feature value e and $C(k)$ is the estimated cost of applying k.

The EDC of a knowledge state, then, is the smallest EPC of the knowledge sources that can be executed at that state:

$$EDC(n) = \min_{k \in KS(n)} (EPC(n,k))$$

where $KS(n)$ is the set of VKSs applicable at node n.

The equations above establish a mutually recursive definition of the expected decision cost of a knowledge state. The EDC of a knowledge state is the EPC of the optimal VKS application at the state; the EPC of a VKS application is the expected cost of applying the VKS plus the expected remaining EDC after the VKS has been applied. The recursion bottoms out at terminal nodes, whose EDC is zero. Since every path through the object recognition graph ends at either a subgoal or a failure node, the recursion is well defined. Furthermore, since the EDC of a level's start state estimates the expected cost of verifying a hypothesis at that level of abstraction, the EDCs of all the start states can be combined with estimates of the number of hypotheses generated at each level to estimate the expected run-time of the strategy as a whole.

3.4 Experimental Results

The previous sections give a simplified description of a complex system that has only recently been implemented. Because the system is new, complete and thorough experiments testing its success both as a knowledge engineering tool and as a machine learning system are only now underway; in this section we report the results of one such experiment.

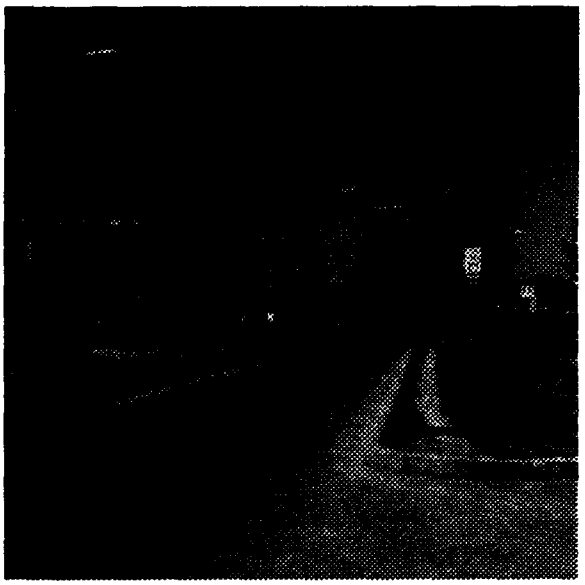


Figure 3: The first of twenty pictures taken of the Marcus Engineering building.

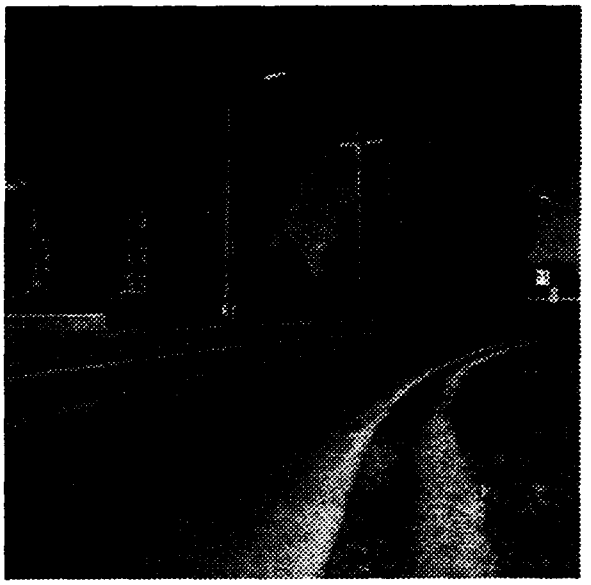


Figure 4: The last Marcus Engineering image

The goal of the experiment was to test SLS within the scenario of learning to accurately recognize the pose of a complex object from an approximately known viewpoint; other experiments are testing its ability to perform 2D recognition and to perform 3D recognition from arbitrary viewpoints. The image data for the experiment were twenty images of the Marcus Engineering building on the UMass campus like the ones shown in Figures 3 and 4, taken along a dirt path at distances ranging from three to four hundred feet from the building. The pictures were taken with the image's y-axis parallel to gravity (i.e. with zero tilt and roll), however there were still rotations (pan) from one image to the next, so that the pose of the building has four free parameters, three locational and one rotational. The goal was to learn a strategy that could identify the pose of the building to within 10° rotation (pan), 5% depth (scale) and 1° of the correct image angle (the angle from the focal point of the camera to the object).

The knowledge sources available to SLS included a geometric matcher for comparing wire-frame models to image data ([1]), perspective analysis routines for estimating orientations ([4, 10]), a line grouping system ([13]), a pattern classification technique ([2]) and template matching routines. It was also given knowledge sources for checking domain constraints such as distance from an object to the camera or the height of an object above (or below) the camera plane.

SLS was tested by a "leave one out" scheme in which strategies were trained on nineteen images and tested on the twentieth. SLS strategies were able to generate correct 3D pose hypotheses in nineteen of the twenty tests (in the twentieth case it generated no pose hypotheses at all). It was easier to generate goal-level hypotheses than to verify them, however, since none of the available VKSs were sensitive to small changes in rotation or scale. At the goal level, strategies compute probabilities of correctness by sampling training hypotheses rather than absolutely accepting or rejecting each hypothesis; Table 1 shows the errors in pan, scale and image angle for the most probable pose returned for each image, while Figure 5 shows the projection of the most likely pose found for the image in Figure 3. (When several pose hypotheses were tied for the highest confidence value we averaged the absolute values of their errors in Table 1). In sixteen of the twenty tests, the most probable pose was within the tolerance thresholds set by the user. In three of the tests, the most probable pose was just outside the thresholds, by either a fraction of a degree in pan (10.11° for image 2, 10.39° for image 16) or a percent in scale (6.06% for image 20). As stated above, no poses at all were generated in the twentieth case (image 6).

4 Conclusion

It is generally accepted that the timely and appropriate use of relevant knowledge can substantially reduce the search encountered in establishing 'instance-of' relationships between image data and its interpretation(s). Unfortunately, the problem of how to acquire and structure knowledge has limited most knowledge-based vision systems to highly constrained domains. The schema learning system (SLS) is an experimental system that learns how to recognize objects from training images. At the moment, SLS still requires a human to supply it with pa-

#	Prob	Pan(°)	Scale(%)	ImAngle(°)
1	0.78	4.95	1.18	0.13
2	0.82	10.11	2.08	0.03
3	0.76	3.83	0.74	0.07
4	0.77	1.76	1.76	0.14
5	0.77	4.92	4.16	0.13
6	–	–	–	–
7	0.76	4.17	2.42	0.21
8	0.76	2.95	0.60	0.09
9	0.77	1.54	2.07	0.04
10	0.77	4.48	2.24	0.25
11	0.79	3.27	4.72	0.08
12	0.76	2.88	1.73	0.05
13	0.78	3.58	4.01	0.20
14	0.78	0.07	2.33	0.08
15	0.78	1.16	2.84	0.05
16	0.88	10.39	0.32	0.15
17	0.77	2.33	2.64	0.10
18	0.78	1.40	1.95	0.13
19	0.79	1.33	4.25	0.12
20	1.00	9.62	6.06	0.16

Table 1: The errors between the most probable pose hypothesis and the true pose for each test image. Pan refers to difference in rotation about the gravitational axis measured in degrees, scale to the distance from the camera to the object measured as a percentage of the true distance, and image angle to the difference in angle between the rays from the camera to the object. The user's tolerance thresholds were 5% scale, 10° pan and 1° image angle.

rameterized knowledge sources and a set of interpreted training images. The previously time-consuming process of supplying control knowledge, however, has been replaced by a system that automatically learns control strategies that minimize the expected cost of recognition.

References

[1] J.R. Beveridge, R. Weiss and E.M. Riseman. "Optimization of 2-Dimensional Model Matching," *Proc. of the DARPA Image Understanding Workshop*, Palo Alto, CA., June 1989. Pp. 815-830.

[2] Carla Brodley and Paul Utgoff. "Linear Machine Decision Trees," submitted to *Machine Learning*.

[3] J. Brolio, B.A. Draper, J.R. Beveridge and A.R. Hanson. "The ISR: an intermediate-level database for computer vision", *Computer*, 22(12):22-30.

[4] Robert T. Collins and Richard S. Weiss. "Vanishing Point Calculation as a Statistical Inference on the Unit Sphere." *3rd. International Conference on Computer Vision*, Osaka, Japan, 1990.

[5] B.A. Draper, R.T. Collins, J. Brolio, A.R. Hanson and E.M. Riseman. "The Schema System," *International Journal of Computer Vision*, 2:209–250.

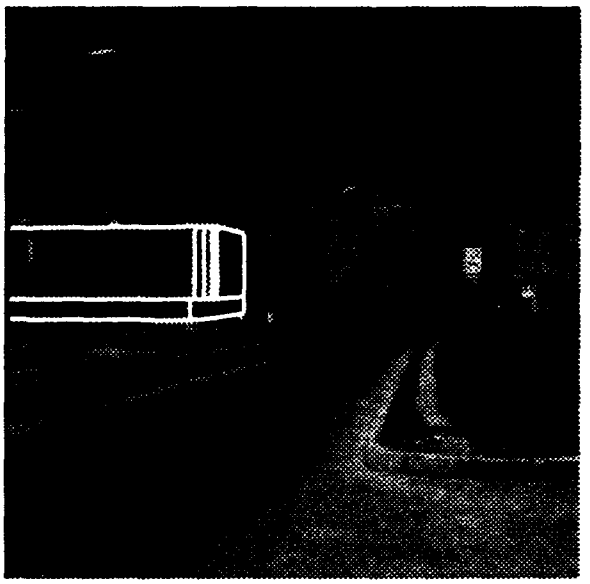


Figure 5: The most probable pose generated for the image shown in Figure 3 (Image #1 in Table 1).

[6] B.A. Draper and E.M. Riseman. "Learning 3D Object Recognition Strategies", *3rd International Conference on Computer Vision*, Osaka, Japan, Dec., 1990. Pp. 320-324.

[7] B.A. Draper. *Learning Object Recognition Strategies*, forthcoming Ph.D. dissertation, Univ. of Massachusetts.

[8] A.R. Hanson and E.M. Riseman. "VISIONS: A Computer System for Interpreting Scenes," in *Computer Vision Systems*, Hanson and Riseman (eds.), Academic Press, N.Y., 1978. Pp. 303-333.

[9] A.R. Hanson and E.M. Riseman, "The VISIONS image understanding system – 1986" in *Advances in Computer Vision*, C. Brown (ed.), Erlbaum: Hillsdale, N.J., 1987.

[10] Ken-ichi Kanatani. "Constraints on Length and Angle," *Computer Vision, Graphics, and Image Processing*, 41:28-42.

[11] T. Matsuyama. "Expert Systems for Image Processing: Knowledge-Based Composition of Image Analysis Processes" *Computer Vision, Graphics, and Image Processing*, 48:22–49.

[12] D.M. McKeown, W.A. Harvey, and L.E. Wixson. "Automating Knowledge Acquisition for Aerial Image Interpretation" *Computer Vision, Graphics, and Image Processing*, 47:37–81.

[13] George Reynolds and J. Ross Beveridge. "Searching for Geometric Structure in Images of Natural Scenes," *Proc. of the DARPA Image Understanding Workshop*, Los Angeles, CA., Feb. 1987. pp. 257-271.

Sensor Planning in an Active Robotic Work Cell

Steven Abrams and Peter K. Allen*
Center for Research in Intelligent Systems
Computer Science Department
Columbia University
New York, NY 10027

Abstract

In this paper, we discuss techniques for extending the sensor planning capabilities of the "MVP" (Machine Vision Planning) system to include motion in a well-known environment. In a typical work cell, vision sensors are needed to monitor a task and provide feedback to motion control programs or to assess task completion or failure. In planning sensor locations and parameters for such a work-cell, all motion in the environment must be taken into account in order to avoid occlusions of desired features by moving objects and, in the case where the features to be monitored are being manipulated by the robot, to insure that the features are always within the camera's view. Several different sensor locations (or a single, movable sensor) may be required in order to view the features of interest during the course of the task. The goal is to minimize the number of sensors (or to minimize the motion of the single sensor) while guaranteeing a robust view at all times during the task, where a robust view is one which is unobstructed, in focus, and sufficiently magnified. In the past, sensor planning techniques have primarily focused on static environments. We present techniques which we have been exploring to include knowledge of motion in the sensor planning problem. Possible directions for future research are also presented.

1 Introduction

Recently, there has been much research in the field of sensor planning [1, 2, 4, 5, 7, 8, 11, 12]. The basic problem is that in setting up an automated system for monitoring some process, the effectiveness of the system can largely be determined by the locations, types and configurations of the sensors used. To manually determine these parameters on a case by case basis may not be cost efficient or accurate, and the resulting system may not be optimal in any sense. It may be better to have an automated system for determining the sensor locations and parameters for monitoring a given task.

To that end, many systems have been and are being developed which, based on geometric models of an environment and models of the sensors, can generate sensor locations and settings which provide a robust view of specific features so that the features are detectable, recognizable, measurable, or meet some other task constraints. In general, the sensors are cameras and a robust view implies that the camera must have an unobstructed view of the entire feature set, which must lie within the depth-of-field of the camera and must be magnified to a given specification. Sensor planning systems can then generate camera locations, orientations, lens settings (focus-ring adjustment, focal length, aperture), and in some cases lighting plans to insure a robust view of the features [10].

Most of the methods presented to date have only addressed static environments such as would be found in a post-manufacturing inspection task (an exception is the VIO system of Niepold et. al. in [7]). The approach taken is to perform an off-line analysis of the geometric and optical constraints for a static environment and, via a generate-and-test or a synthesis approach, give one or more sensor location and parameter settings which are valid only for the specific static environment which was analyzed. When objects in the environment need to be moved for some reason, new sensor locations need to be computed off-line. This works well for quality-control or inspection tasks where, for example, parts can be fed to a specific location and orientation in the environment for inspection.

There are many instances where moving scenes

*This work was supported in part by DARPA contract N00039-84-C-0165, NSF grants DMC-86-05065, DCI-86-08845, CCR-86-12709, IRI-86-57151, IRI-88-1319, North American Philips Laboratories, Siemens Corporation and Rockwell International.

may need to be monitored. Manufacturing or assembly tasks can be visually guided or monitored. Telerobotic operations need some form of sensor feedback. For these and other tasks, the sensor planning techniques presented to date would be inadequate due to the dynamic nature of the environments. But for these applications, it would still be better to have the robotic system remain in control of the sensor positions and settings to insure reliable monitoring rather than to require manual control over the sensors.

In this paper, we describe our most recent research in extending the MVP (Machine Vision Planning) System [11, 12] to plan sensor locations and settings for a changing environment. After a brief overview of the MVP system, the remainder of this paper will focus on an explanation of the constraints we impose on the environment, the theory behind our method, and a technique which helps to incorporate temporal reasoning into spatial problems. We also present examples which show this temporal reasoning working in conjunction with the MVP system. Finally, we present an overview of future work in this area, aimed at strengthening our ability to reason temporally as well as spatially for sensor planning and other problems in robotics.

2 Overview of MVP

A complete description of the MVP system is beyond the scope of this paper. In brief, MVP takes a constraint based description of the vision task requirements and synthesizes what has been termed a *generalized viewpoint*, which is an eight-dimensional vector incorporating sensor location, orientation, and lens parameters including aperture and effective focal length. The constraints MVP considers in determining viewpoints are depth-of-field, field-of-view, resolution, and unoccluded visibility [11, 12].

MVP contains analytical relationships for the optical task constraints (resolution, focus, field-of-view), and uses 3-D solid geometric models of the environment to formulate visibility constraints. (The geometric models are limited to general polyhedra, both convex and concave; curved surfaces are not permitted.) The constraint equations can be thought of as defining hypersurfaces bounding feasible regions in the 8-dimensional parameter space of the generalized viewpoint. These constraints are combined in an optimization setting to produce a generalized viewpoint which meets all task constraints with as much margin for error in sensor placement and setting as possible (i.e., as far away from all hypersurfaces as possible). Figures 1 and 2 show how MVP, from a CAD description of the object to be viewed and its environment, can generate the visibility region for viewing the desired features. This region is calculated to be the total volume in space from which the features are viewable without obstruction. This volume is used in the optimization stage of MVP for finding the best viewpoint.

While MVP's synthesis approach makes it better suited to the extensions we present here than other less analytical approaches, there is no reason why the temporal extensions discussed here can not be applied to other sensor planning algorithms. In fact, it is our hope that these methods can be applied to 3-D planning problems in general.

3 The Introduction of Motion

In sensor planning problems, there is normally a well-defined set of target features which need to be monitored. These might correspond to a section of a part which has just come off the assembly line which the vision system might need to examine for defects. In an active environment, the feature set might correspond to a section of a larger assembly being operated on, which we might need to monitor during the operation. For this work, we restrict ourselves to consider only the motion of obstacles in the environment, and not the motion of the target. We also assume complete knowledge of the environment in the form of 3-D geometric models. Finally, we assume knowledge of the motion of the obstacles in advance; unplanned motion may not take place.

The most important result of these limitations is that once we have a viewpoint[1] which is valid for a given instant in time t_n, it is guaranteed to be valid with respect to all *optical* constraints for all times t_m for $m > n$. This is fairly simple to show. Once we have a properly magnified and focused view, if neither the target nor the camera move, the object remains in proper focus with an unvarying magnification. The movement of obstacles means that only occlusion needs to be detected at later times.

One other point worth mentioning is that the static sensor planning problem does not require the camera to be attached to a robot. For the dynamic sensor planning problem, the camera must be movable from one viewpoint to another during the course of the task being monitored in order to maintain a robust view of the target.

[1] Here, and elsewhere in this paper, when we refer to a *viewpoint* we are actually referring to the *generalized* viewpoint mentioned earlier.

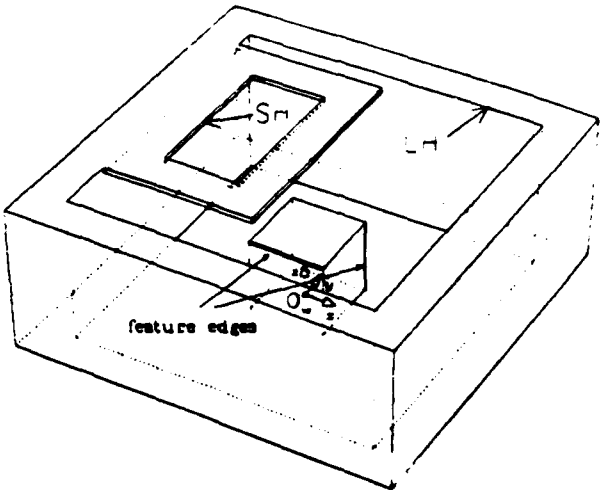


Figure 1: CAD model of the object (from [11]).

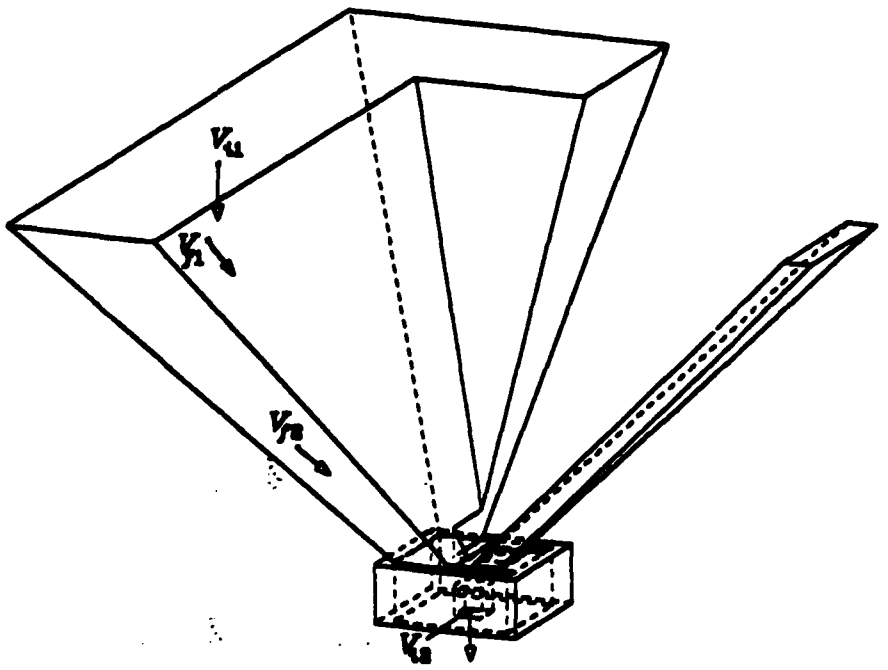


Figure 2: The Visibility region for both edge features (from [11]).

To summarize, the exact problem we are dealing with is one in which an accurately movable camera is being used to monitor a task. In this task, the actual target we are monitoring does not move, but other objects in the environment, such as a robot arm, or other mechanical parts, move in a way which is known *a priori*. The problem is to find where to place the camera, and when and where to move the camera, so that at all times during the task, we have a good viewpoint for monitoring the task.

As an example, a spot-welding robot may be working on a stationary object (such as a car body) in a factory. There might be a vision system used to monitor the weld for defects and for accuracy. The sensor would need to be placed in such a way as to avoid occlusion by the many moving obstacles in the environment, such as the welding arm itself and any moving peripherals needed for this task. The target itself, that is the area to be welded, would remain stationary for this task.

4 Relation to Motion Planning

There is a very close relationship between the problem of sensor planning with moving obstacles and that of motion planning with moving obstacles. The essence of both problems is to find a representation which relates the temporal and spatial aspects of an object's motion. The essential difference is that while path planning needs to find a path which moves through a time-varying environment to reach a goal without hitting any obstacles, sensor-planning is searching for one or more generalized viewpoints which remain valid (unoccluded) for the duration of the task. The implications of this difference are that we can not envision planning a point that travels through configuration space or configuration space-time [3, 6]. This is because our "point" is actually a cone with its apex at the sensor and its base at the target polygon. We need to detect when obstacles will breach this viewing cone, not just detect the collision of a single point with a configuration space obstacle. The second difference is that in motion planning, we search for an unobstructed path. In sensor planning, the preference is to remain stationary. We are in search of a single viewpoint, if possible, which is valid during the entire operation, or, if that is not possible, as few viewpoints as possible.

5 The Naive Approach

One way to handle added complexities to a problem is to ignore them until they become an issue. Taking this approach to motion in sensor planning yields the following naive algorithm:

1. Compute a viewpoint for the initial state of the system, considering all obstacles in the environment as they are before any motion takes place.
2. At every time interval Δt, test the current viewpoint against the model of the changed environment.
3. If, at some instant t_n, the viewpoint is found to be invalid due to the movement of obstacles, compute a new viewpoint based on the current state of the model, and go back to step 2.

The algorithm can be run in advance, off-line, since the changes the environment goes through over time are known, so it would not even be necessary to have a function which evaluates or generates viewpoints in real time. The entire problem can be simulated, and the time intervals where the sensor needs to be moved and reset can be noted. During the process, the robotic system can pause, reposition the camera, and then continue its operation.

This approach has several major drawbacks:

- This algorithm is clearly not optimal in its use of the viewpoint evaluation function. In a task which takes $M \times \Delta t$ time to complete, the viewpoint needs to be evaluated exactly M times.
- The algorithm makes no attempt to reduce the number of sensor replacements required.
- A viewpoint is used up until the moment *after* it has become invalid, or at least up until the point at which the margin for error becomes very small. As an example, say the algorithm determines that at time t_n, the initial viewpoint is no longer acceptable. Although the initial selection of a viewpoint was chosen to have a large margin for error, this error margin only existed at time t_0; at some time before t_n, due to errors in sensor placement, etc, the viewpoint may be bad.
- The accuracy of this method is dependent upon the time interval Δt used. Δt is essentially the sampling frequency with which we test the environment. The viewpoint may become invalid between two samples and yet be valid at each sample. This behavior is not desirable.

- Knowledge of future positions of the obstacles is not used in this method. Given that the planning and evaluation functions are fast enough, there is no benefit to computing any information in advance. It is conceivable that at each time interval t_n, when we plan a new viewpoint, the very next motion (or the sequence of motions in the near future) make the new viewpoint invalid very quickly, perhaps even at time t_{n+1}. This is very bad; the sensor may need to be repositioned far to frequently.

The problem specification indicates that we have more information at our disposal than this solution uses. By using all available knowledge, that is, our knowledge of the temporal as well as the spatial aspects of the environment, we can hope to generate viewpoints which are valid (by a larger margin) for longer time intervals.

6 Temporal Considerations

In order to make use of our knowledge of motion and time in the environment, we define a concept which relates the geometric orientation of an object, and its motion through space over a given time interval. The structure which embodies this relationship for a given object over a given time period is called a *minimal temporal object.* To illustrate this concept, note the polyhedron in figure 3. It is shown with a vector indicating its linear trajectory.

Definition 1 *A* minimal temporal object *is the set of all points through which a given object O passes during its motion over a given time interval T. The minimal temporal object representing the motion of O over T is notated as $\hat{\mathcal{T}}(T,O)$.*

Figure 4 shows the minimal temporal object for the polyhedron in figure 3 moving in the direction indicated. For a given object, and its trajectory and velocity over a given time period, there is one unique $\hat{\mathcal{T}}(T,O)$. The task of computing $\hat{\mathcal{T}}(T,O)$ is equivalent to computing the complete volume swept by O during its motion over T. For general objects moving in arbitrary paths, $\hat{\mathcal{T}}(T,O)$ may be exceedingly expensive to compute. This is why, in working within the sensor planning framework, we have been dealing with approximations to $\hat{\mathcal{T}}(T,O)$. We restrict ourselves to approximations of $\hat{\mathcal{T}}(T,O)$ which meet the following definition:

Definition 2 *A* temporal object *is defined as any volume which contains the set of all points through which a given object O passes during its motion over a given time interval T. A temporal object representing the motion of O over time interval T is notated as $\mathcal{T}(T,O)$.*

The most important consequence of this definition is that $\mathcal{T}(T,O)$ is necessarily contained within $\hat{\mathcal{T}}(T,O)$. Rough methods can be developed to compute $\mathcal{T}(T,O)$ and use them as approximations to $\hat{\mathcal{T}}(T,O)$ for planning. Note, that for any object moving through a given path, $\mathcal{T}(T,O)$ is not unique while $\hat{\mathcal{T}}(T,O)$ is.

Due to a result by Weld and Leu in [13], the volume formed by sweeping a polyhedral object along an arbitrary path is equivalent to the volume formed by sweeping each face along the same path and unioning these swept volumes together. The expensive portion of this algorithm (in the case of translational motion only) is in the unioning. A polyhedron of n faces requires n boolean unions to be swept via this method. Since, in the *Temporal Sensor Planning* algorithm (which follows), it may be necessary to compute swept volumes often, we have opted for a faster method of computing a $\mathcal{T}(T,O)$ as opposed to an accurate, but slower method for computing $\hat{\mathcal{T}}(T,O)$.

We use a simple algorithm to compute a $\mathcal{T}(T,O)$ given that T moves through a linear path. This is not particularly restrictive, since any arbitrary path can be approximated by a piecewise linear path, and the algorithm can be repeated over each linear segment of a path to compute the $\mathcal{T}(T,O)$ for the whole interval. Here, we describe the algorithm for computing $\mathcal{T}(T,O)$ when O is a polyhedron known to move in a linear path of length n in the $\vec{v}$ direction.

Generation of $\mathcal{T}(T,O)$

1. Calculate plane $\mathcal{P}$, which is the plane defined by the unit vector $\vec{v}$ and the point p of O in the extreme $-\vec{v}$ direction. That is, as O moves in the $\vec{v}$ direction, p is the point "furthest back," and $\mathcal{P}$ is the plane perpendicular to the trajectory of O, and containing p.
2. Project all vertices of O onto the $\mathcal{P}$ and take the convex hull of these points, creating a polygon s on $\mathcal{P}$.
3. $\mathcal{T}(T,O)$ is the right generalized cylinder with a linear axis parallel to $\vec{v}$, a constant cross-section, with a base on s and a height of n plus h, where h is the overall length of O in the $\vec{v}$ direction.

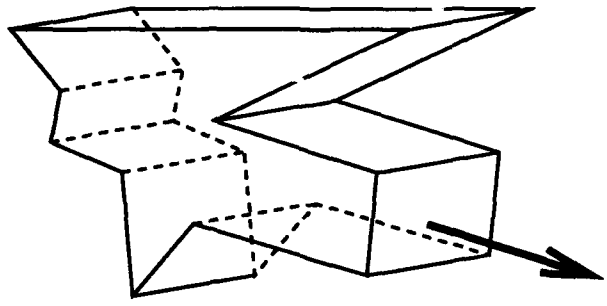


Figure 3: Polyhedron floating in space with direction vector

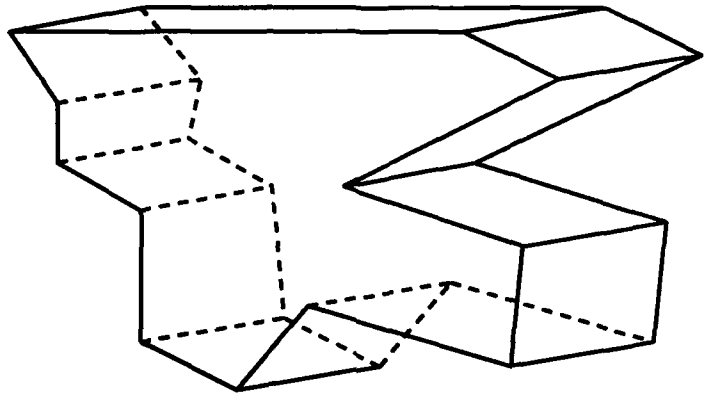


Figure 4: The unique minimal temporal object

This algorithm is illustrated in figure 5.

Theorem 1 *Any volume created by the above algorithm necessarily contains* $\hat{\mathcal{T}}(T, O)$.

Proof:

1. By step 3 above, all perpendicular cross sections of $\mathcal{T}(T, O)$ are the same.

2. By step 2, the cross sections must be polygons wholly containing any perpendicular cross section of $\hat{\mathcal{T}}(T, O)$.

3. The overall length of $\mathcal{T}(T, O)$ is at least as long as that of $\hat{\mathcal{T}}(T, O)$ by the final step.

4. These three conditions show the total inclusion of $\hat{\mathcal{T}}(T, O)$ by $\mathcal{T}(T, O)$.

The key to using a temporal objects for sensor planning (or, in fact, for any collision avoidance problem) is that in planning around an obstacle given by $\mathcal{T}(T, O)$, you guarantee that you have avoided the actual obstacle O at any instant in interval T.

7 Temporal Objects in Sensor Planning

The essence of our approach to sensor planning around moving objects is to plan around the temporal objects generated from the original objects and their motion. The temporal objects, once calculated as regions in space, are treated as stationary objects for a static planning problem. In many cases, this is too restrictive. There may be one viewpoint which is valid for one portion of the time interval, and another viewpoint which is valid for another portion, yet there is no viewpoint which is valid for the entire interval. In cases such as this, our algorithm subdivides the interval in half whenever faced with a failure, and replans for the two halves independently.

More formally, this algorithm is described as follows. Assume we have a polygonal target τ which we wish to monitor during the time interval $T = [t_0, t_n]$. During T, there is a set of known obstacles O_0 through O_m, which move in known paths. The goal is to plan a single viewpoint valid for the entire interval, if such a point exists, or to determine a sequence of viewpoints.

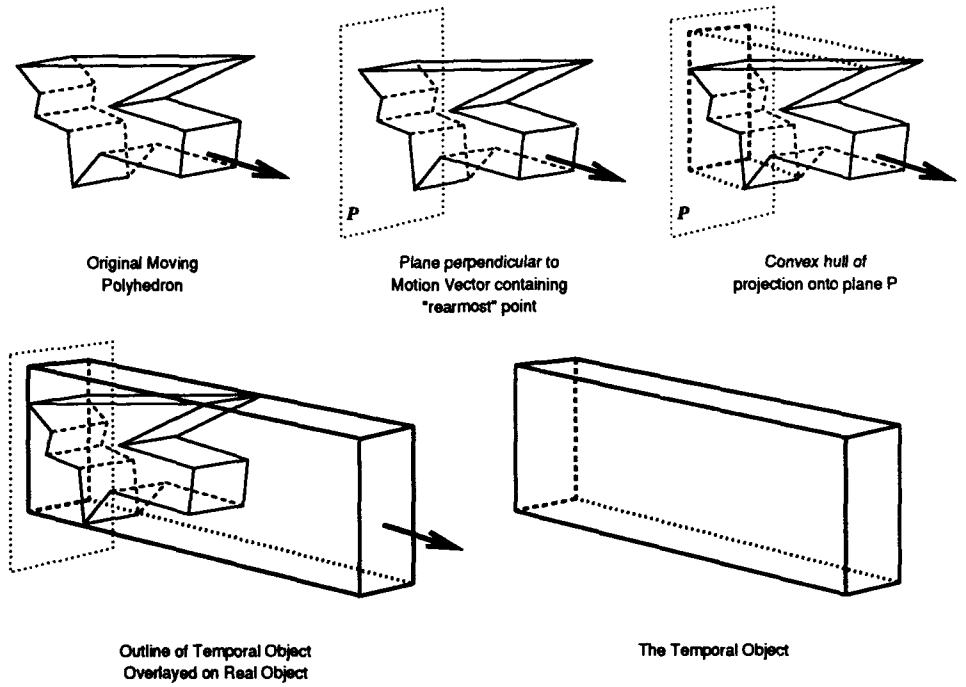


Figure 5: Generation of a Temporal Object

Temporal Sensor Planning (*TSP*)

1. Compute $\mathcal{T}(T, O_i)$ for each of the m obstacles.
2. Use MVP to compute a viewpoint using the set of temporal objects spanning time interval T as the potential occluding bodies.
3. If MVP can successfully find a viewpoint which is valid in the presence of all of the temporal obstacles, it is guaranteed to be valid for any instant in T. If such a point is found with MVP, the algorithm terminates with a successful result: it has planned a single viewpoint.
4. If no such viewpoint is obtainable, we divide T into $T_1 = [t_0, t_{n/2}]$ and $T_2 = [t_{n/2}, t_n]$, and run the Temporal Sensor Planning algorithm on each subinterval.

The binary partitioning of the last step continues until we have found a valid viewpoint for all of T, or until we have divided into time intervals of some minimal preset length ϵ. If sub-intervals too small are reached, the determination is that the motion is too complex for this method to provide meaningful results. Typically, one chooses ϵ to be large enough so that it is feasible to stop and reset the sensor every ϵ interval, since, in the worst case, that is what might happen. If the *TSP* algorithm determines that the motion in the workcell is too complex to plan viewpoints, it might be an indication that the activity in the workcell is too complex to be monitored and that the task itself should be replanned.

It is important to note that in step 2 of the *TSP* algorithm, we rely on MVP to identify the fact that it has been unable to to find a valid sensor viewpoint. MVP itself relies on a nonlinear constrained optimization to compute a viewpoint. The failure of MVP to find a viewpoint does not guarantee that one does not exist. However, when a valid viewpoint is so well-hidden within the hypersurfaces of 8-dimensional parameter space that our optimization routine can not locate it, there is a very good chance that the viewpoint does not provide much of a margin for error. Therefore MVP's failure to find a viewpoint, while not a guarantee of the nonexistence of a *valid* viewpoint, is an excellent indication that there is no *good* viewpoint.

8 Experimental Results

We have implemented *TSP* in conjunction with the MVP system, and have produced simulated results showing the effectiveness of *TSP* as a method of extending 3-dimensional planning algorithms to include the time domain. While only the visibility constraint of MVP is of concern to *TSP*, it is important to note that the other constraints do play a role in whether or not *TSP* can find a valid viewpoint. In these experiments, the sensor modeled is

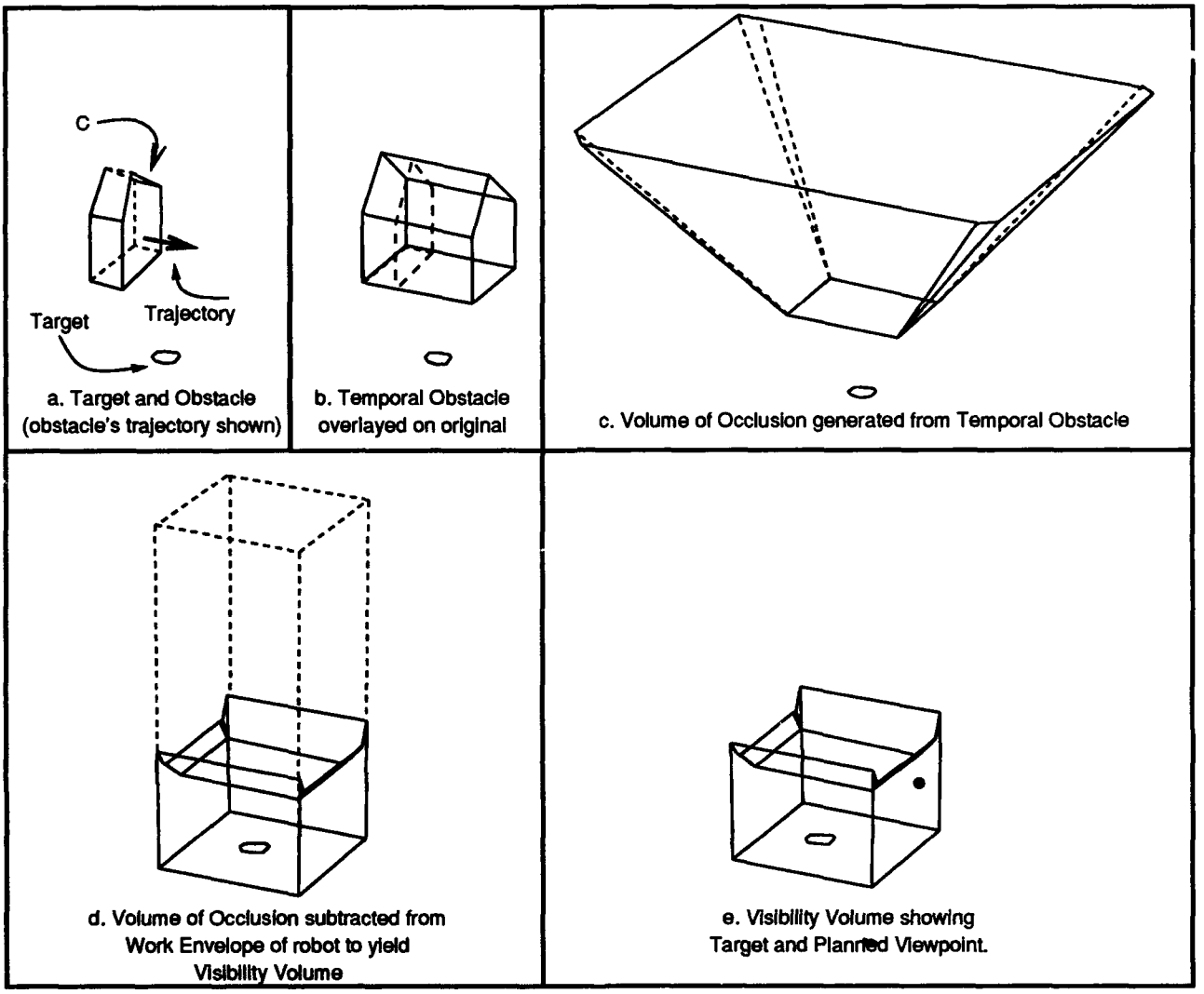


Figure 6: Simulation of *TSP* and MVP

a typical CCD camera, and a resolution constraint of one pixel per 0.1 inches was used. The geometric models of the objects and environment as well as those of the temporal objects were calculated using the ACIS solid modeler [9].

In the first example, a polyhedral object C is the obstacle which moves in a linear trajectory above an octagonal target at a constant velocity over time interval T (figure 6a). Next, using the algorithm presented earlier, $\mathcal{T}(T, C)$ is generated (figure 6b). The corresponding volume of occlusion is also generated using the methods presented in [12] (figure 6c). Note that this volume represents an overapproximation to the volume in space from which, at some point in the interval T, the view of the target would be blocked due to C.

The volume of occlusion is subtracted from the work envelope of the robot carrying the camera to yield the visibility volume (figure 6d). Note that any point in this volume is now guaranteed to have an occlusion-free view of the target region for the entire time interval T. This volume is used in the optimization portion of MVP to generate the shown viewpoint which is clearly valid for the entire motion of C. Note that in Figure 6e, the planned viewpoint is clearly within the visibility volume; the line of sight from the viewpoint to all points on the target does not violate the temporal obstacle.

In the next example, we lowered C so that MVP would be unable to find a viewpoint which maintained both visibility and focus over the entire time interval. The partition of the time interval produced two sub-intervals, both of which had valid viewpoints easily computed using MVP (see Figure 7).

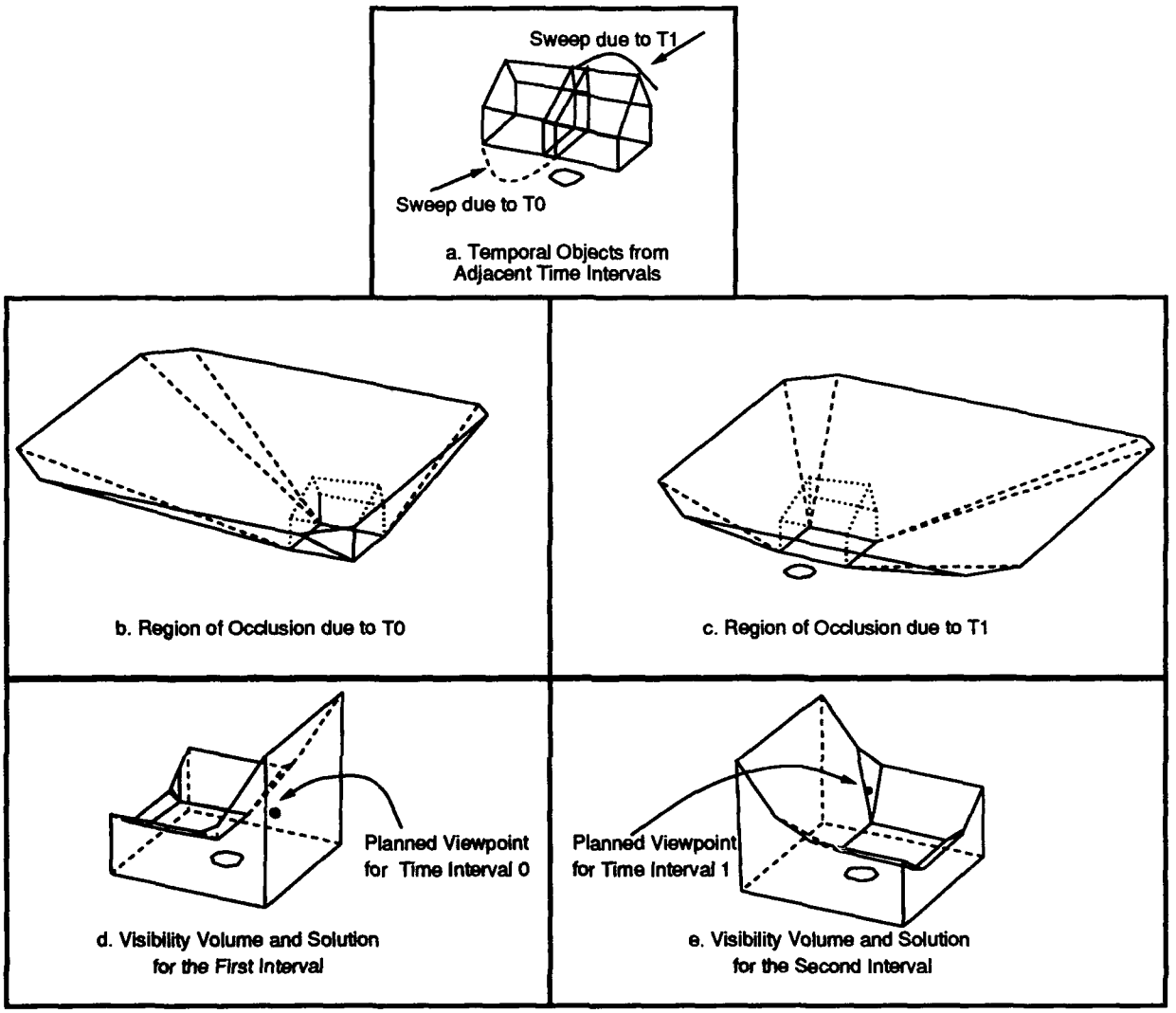


Figure 7: Simulation of *TSP* and MVP showing Binary Partition

9 Conclusions

This paper has presented two main ideas. First, we have presented work in extending geometric planning problems to include time and motion. The use of the temporal object makes the temporal components of the problem invisible to any underlying geometric solver. This makes it a very useful notion; many other geometric planning problems may benefit from this concept.

Second, we have successfully extended our MVP system to plan sensor locations in a time-varying environment. This is notable in that to the best of our knowledge, motion has not been widely addressed in the sensor planning literature. This particular planning task is quite important to us and we plan on concentrating future work on improving the performance of MVP under various conditions, especially under more weakly constrained motion conditions.

For example, while the methods outlined above work well for the translational motion of known obstacles, they have yet to be used to model the motion of the target as well as the obstacles. The fact that we have constrained ourselves to moving obstacles has allowed us to ignore the effects of motion on all constraints other than visibility. Once we allow the target to move, resolution, focus, and field-of-view constraints must be dealt with in an environment which changes over time. Generating a temporal object for the target seems to be a reasonable approach, but it may not be sufficient. The problem of a moving target is inherently more complex than that of a moving obstacle.

Note that in sweeping a polyhedral obstacle along

a linear trajectory, as we have done, the result is another polyhedral obstacle. Now note that a target consisting of a single point, when swept along a linear path, yields a line; a linear target swept along a linear path yields a polygon; a polygonal target swept linearly yield a polyhedra. Future work will explore how the planning algorithm can plan to view this temporal target, which is one dimension higher than the original target.

There may be other methods which may be used to plan sensor viewpoints around moving obstacles which take advantage of the fact that the visibility volume is the only constraint which changes through time. We may be able to examine how the actual obstacles move in relation to the current sensor's line-of-sight to the target to determine when replacement may be needed, for example.

Allowing motion in the sensor planning problem opens it up to many more subproblems which we hope to explore in future work. For example, it may be desirable to examine ways of ensuring that the series of viewpoints lay along a path which is accessible. We may also be exploring ways of generating a continuous path along which to move the sensor which maintain a particular view of a target.

References

[1] C. I. Connolly. The determination of next best views. In *Proceedings 1985 IEEE International Conference on Robotics and Automation*, 1985.

[2] C. K. Cowan and A. Bergman. Model based synthesis of sensor locations. In *Proceedings 1989 IEEE International Conference on Robotics and Automation*, December 1989.

[3] M. Erdmann and T. Lozano-Perez. On multiple mpoving objects. *Algorithmica*, 2(4):477–521, 1987.

[4] S. A. Hutchinson and A. C. Kak. Planning sensing strategies in robot work cell with multi-sensor capabilities. In *Proceedings 1989 IEEE International Conference on Robotics and Automation*, December 1989.

[5] K. Ikeuchi and T. Kanade. Modeling sensors: Towards automatic generation of object recognition programs. *Computer Vision, Graphics, and Image Processing*, 48:50–79, 1989.

[6] T. Lozano-Perez. Spatial planning: a configuration space approach. *IEEE Transactions on Computers*, 32(2):108–120, 1983.

[7] R. Niepold, S. Sakane, and Y. Shirai. Vision sensor set-up planning for a hand-eye system using environmental models. In *Proceeding Soc. Instrum. Control Eng. Japan*, Hiroshima, Japan, July 1987.

[8] S. Sakane, M. Ishii, and M. Kakiura. Occlusion avoidance of visual sensors based on a hand eye action simulator system: HEAVEN. *Advanced Robotics*, 2(2):149–165, 1987.

[9] Spatial Technology Inc. *ACIS Kernel Interface Guide.*

[10] Konstantinos Tarabanis. Sensor planning in computer vision: A review. Technical report, Columbia University Department of Computer Science, 1991.

[11] Konstantinos Tarabanis, Roger Y. Tsai, and Steven Abrams. Planning viewpoints that simultaneously satisfy several feature detectability constraints for robotic vision. In *Proceedings Fifth Inernational Conference of Advanced Robotics*, 1991.

[12] Kostantinos Tarabanis, Roger Y. Tsai, and Peter K. Allen. Automated sensor planning and modeling for robotic vision tasks. In *Proceedings 1991 IEEE International Conference on Robotics and Automation*, April 1991.

[13] John D. Weld and Ming C. Leu. Geometric representation of swept volumes with application to polyhedral objects. *International Journal of Robotics Research*, 9(5):105–117, October 1990.

Subsection H:

Robotics Applications

Planning for Perception in Robot Driving

Douglas A. Reece and Steven A. Shafer
October, 1991

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

As mobile robots attempt more difficult tasks in more complex environments, they are faced with combinatorially harder perceptual problems. In fact, computation costs for perception can easily dominate the costs for planning in a mobile robot. Existing perception systems on mobile robots are potentially many orders of magnitude too slow for real-world domains. In this paper we show that active vision at the system level can make perception more tractable. We describe how our planning system for a complex domain, tactical driving, makes specific perceptual requests to find objects of interest. The perception system then scans the scene using routines to search for these objects in limited areas. This selective vision is based on an understanding and analysis of the driving task. We illustrate the effectiveness of request-driven routines by comparing the computational cost of general scene analysis with that of selective vision in simulated driving situations.

1. Introduction

Today's mobile robots can drive down hallways and roads and across fields without getting stuck or colliding with obstacles. As these r bots attempt more challenging tasks, they will need better perception and planning systems. In the past, planning and perception have been independent components of the robot reasoning system. Figure 1-1 shows that in traditional systems the planning component works on a symbolic world model, which it naively assumes the perception component keeps up to date. The perception component must therefore find and understand *everything* of potential interest to the planner. This arrangement cannot be used in robots in the real world because scenes are too complex and change too fast for unguided, exhaustive interpretation.

Consider the driving scene shown in Figure 1-2. It contains several vehicles. The vehicles are

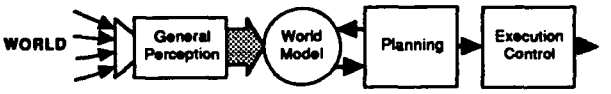


Figure 1-1: A traditional robot control system.

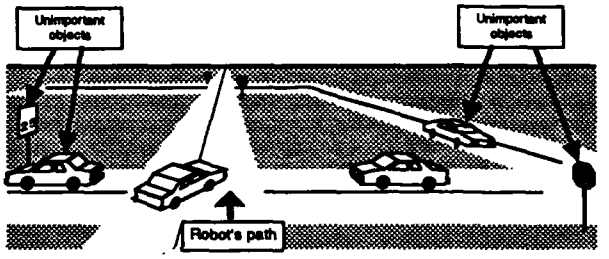


Figure 1-2: A driver can see traffic objects in many places.

different distances and directions from the observer, and so appear in different locations in the image. A general perception system attempting to find all vehicles in the scene would have to search all possible locations, ranges, and poses. For each of these possibilities, the perception system would also have to consider variations in vehicle shape, color, and illumination. The driving scene changes from moment to moment, with vehicles moving, disappearing, reappearing, and becoming partly occluded. The combinatorics of perception in the real world are enormous.

Figure 1-2 also illustrates that even if traffic objects can be found in various poses throughout a scene, it is wasteful to look for them because they are not all important to the robot. We propose to reduce perceptual complexity by using system-level active vision—using the planner to limit what a

robot has to look for, and where it has to look[1]. Figure 1-3 shows the resulting architecture. A robot driver can then use its perception effectively, as shown in Figure 1-4.

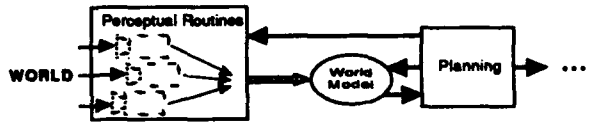


Figure 1-3: An active-vision robot control system.

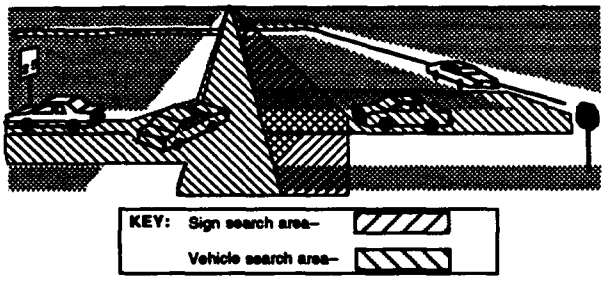


Figure 1-4: Desirable visual search constraints.

In the next section we introduce the domain of driving in traffic, and describe how we define it using an environment simulator and computational model. We then estimate the computational cost of general sensing in the driving domain. Such an estimate demonstrates the futility of a naive approach to perception for a real problem, and provides a basis for evaluating our active vision techniques. Section 3 describes two such techniques. Section 4 illustrates how selective perception techniques work for driving. We discuss how we have implemented our driving model in a computer program so that we can study perceptual issues in simulation. Next we demonstrate how perceptual routines are used in a driving situation. Finally, the section shows the results of simulating a driving situation and compares the computational cost to the naive approach. We conclude in Section 5 that general perception is hopelessly intractable and that selective perception is necessary for a task such as driving.

[1]This paper emphasizes demand-driven perception and routines, but does not address other reductions in search made possible by modeling the world better. Our current work addresses those issues.

2. Driving with General Perception

Driving in traffic is an example of a complex task in a complex, dynamic environment. This paper addresses *tactical* driving [12], which is the selection of speed and steering maneuvers. Tactical driving requires features of both real-time servo-control systems and symbolic reasoning systems. Decisions must be made dynamically in response to changing traffic situations. Tactical driving also involves reasoning about road configurations and traffic control devices to figure out right-of-way puzzles. The environment is visually complex because of the many objects and types of objects—cars, roads, markings, signs, signals, etc. These objects vary in appearance under different conditions. Previous work in autonomous road following and car and sign recognition [3, 4, 5, 8, 10] has shown that perceiving individual traffic objects in constrained situations is computationally expensive; finding all objects in all situations in real time is effectively intractable. In this section we estimate just how difficult this perception problem is.

2.1. Domain Assumptions That Affect Perception

The environment. Driving is a complex human activity that is difficult to describe precisely. However, in order program an autonomous robot to perform such an activity, we must be able to describe exactly what it is the robot has to do. First of all, we must describe the important aspects of the environment. In our research, we did this by developing a model of the driving environment and implementing it in a microscopic traffic simulator called PHAROS [13]. PHAROS contains detailed representations of roads, lanes, intersections, signs, signals, markings, and cars. Although PHAROS provides a rich setting for driving experiments, it also makes important abstractions that simplify and limit the driving problem for our work. For example, all roads are structured with lanes; also, there are no pedestrians or bicylists in PHAROS.

Task requirements. Once we have created a representation of the world, we must specify what it means to drive. Although driving laws are published in books [9], and the driving task has been thoroughly analyzed in isolated situations [11], there are no driving descriptions that actually specify what actions to take at any time. We have developed a computational model of tactical driving called Ulysses [14] that does encode what action to take in any situation in the PHAROS world. Ulysses is a sophisticated model that incorporates knowledge of speed limits, car following, lane

changing, traffic control devices, right of way rules, and simple vehicle dynamics. We use a programmed implementation of Ulysses to drive one vehicle in the PHAROS world. Ulysses is our definition of the driving task for this research.

A general perception system has to find all traffic objects of potential interest to the driving planner. In our driving model, these objects include road regions, road markings, vehicles, signs and signals. By characterizing the size of these objects, we can determine the maximum angular resolution needed to detect them. Traffic objects are characterized in more detail in the technical report version of this paper [15].

Since it is not practical to consider perceiving the entire world, we arbitrarily set a range limit on the system. In some driving situations it may be desirable to see long distances ahead; for example, signals are supposed to be visible from 218m away on a road with 100kph traffic [6, pg. 4B-11], and the sight distance needed for passing is given as 305m at this speed [1, pg. 147]. While Ulysses is capable of driving on simulated highways, for this work we have concentrated on arterial urban streets where the visual environment is more diverse. Since streets have lower speeds than highways, we have chosen 150m as the perceptual range limit.

Sensors. Since the environment is so varied in appearance, we assume that analysis must be based on several types of sensor data. We assume the robot has cameras and laser rangefinders whose images can be registered. This combination is complementary in that a rangefinder is almost immune to the illumination changes across an object which confuse color based segmentation. A camera, on the other hand, can discern markings and other important regions on uniform surfaces like signs [7]. Since the robot must find objects in all directions, we assume it has sensors pointing in several directions. We thus avoid issues such as sensor aiming time. Finally, we assume that the driving robot has sensors with enough resolution to discern the smallest traffic objects at the maximum range. In our model of the environment, the smallest feature is a lane marking that is perpendicular to the line of sight. At 150m, such lines require pixels of about 2.5×10^{-4} degrees.

2.2. Cost Estimates

Interpreting general traffic scenes will be very difficult. We would expect the general perception system to extract many features from the image, including regions, boundaries, lines, corners, etc. The complexity of the environment will in general prevent perception from using single, uniform features to uniquely distinguish objects. Extraction will use intensity, color, optical flow, range, and reflectance data. Features must be grown, characterized, and merged. Scene features will be matched against features of traffic object models to identify traffic objects. This matching will be done in two stages; for example, sign surfaces will first be located before the sign message is examined at higher resolution. Our analysis is explained in detail in [15].

Based on work at CMU on robot driving, and on an examination of the literature, we estimate that the cost of general perception in this domain is

$$Cost \approx 1.1\times10^{4} P + 1.1 P^{2} + 1.7\times10^{-5} P^{3} \quad (1)$$

where P is the number of pixels in the image. The unit of cost is an arithmetic operation on a data value or pixel. The linear term reflects the computations to find colors, edges, optical flow, etc. at each pixel; the squared term comes from pixel clustering and comparing pairs of features to each other; and the cubic term is our estimate of the cost of matching, using constraints to prune the search space. Traffic objects are small enough to require about 1cm resolution, which translates to about 8×10^{7} pixels. The net cost is then

$$\begin{aligned} Cost &\approx 8.9\times10^{11} + 7.2\times10^{15} + 9.0\times10^{18} \\ &\approx 9.0\times10^{18}. \end{aligned}$$

This means that a computer that could perform 10^{8} operations in the 100ms Ulysses decision cycle would be almost 11 *orders of magnitude* too slow to analyze the scene. Even if our estimates are off by several orders of magnitude, it is clear that general perception is intractable.

3. A Model of Active Vision for Driving

3.1. Using Task Knowledge to Constrain Search

We limit visual search in our driving system by having the planner direct the actions of perception. Perception is demand-driven rather than forward-flowing. The driving planner uses task-specific knowledge to determine what objects are relevant to the current driving situation.

Since perception only acts upon the request of the planner, Ulysses starts its decision process with almost no information. At the beginning of each decision cycle, perception finds one key object—the road in front of the robot. Ulysses then applies

appropriate driving knowledge to interpret the situation and generate action constraints. This knowledge generally involves other objects related to the key object. For example, Ulysses looks for cars on and signs next to the road in front of the robot. Thus the program generates demands for perception system to look for these cars and signs. When objects are found—especially objects indicating an intersection—additional rules may become relevant and more objects may need to be found. Ulysses' demands thus incrementally determine *what* the perception system must look for in the scene.

Ulysses also controls visual search by constraining *where* the perception system has to look in the image. As Figure 1-4 shows, not all traffic objects are important to the robot; only the ones with specific relations to the robot are important. For example, the Stop sign in the middle of the figure affects the robot because it is *along the robot's intended path.* The Stop sign to the far right is not. Ulysses can thus specify where to look for objects—not directly in terms of azimuth and elevation ranges, but in terms of previously observed objects. Ulysses conveys this information by making perceptual requests in the form of perceptual routines.

3.2. Perceptual Routines

Perceptual routines are sequences of image processing operations. Ullman describes how visual routines[2] may be used in the human vision system to determine object properties and spatial relations [16]. While the input to such routines is an image processed from the bottom-up, the routines themselves are invoked from the top-down when needed in different tasks. Agre and Chapman used visual routines in their video game-playing system, Pengi [2]. The Pengi planner executed visual routines just like any other actions in order to get information about the world state.

The perception system for Ulysses was inspired by Pengi. As we described above, the rules for driving also have spatial relations incorporated in them. Ulysses uses routines to directly find roads related to the robot, cars related to the roads, signals related to particular intersections, etc. Figure 3-1 lists the routines currently available to Ulysses. The routines in Ulysses assume more domain-dependent processes are available to the routines than is the case in Pengi; for example, the routines must understand how to trace roads instead of just contours, and how to stop scanning when a sign is recognized. Most of the routines in the figure mark objects and locations when they finish, so Ulysses can continue finding objects later. For example, track-lane stops when an intersection is encountered (indicated by the change in lane lines in the well-marked PHAROS world), but marks the intersection so that find-path-in-intersection, find-signal, etc. can find objects relative to that intersection. The next section illustrates how Ulysses uses routines.

Find current lane	Find next car in lane
Mark adjacent lane	Find crossing cars
Track lane	Find next sign
Profile road	Find next overhead sign
Find intersection roads	Find back-facing signs
Find path in intersection	Find signal
Find next lane marking	Marker distance

Figure 3-1: Perceptual routines for Ulysses.

Agre and Chapman point out two advantages of routines in terms of knowledge representation. First, only relevant objects in the world need to be represented explicitly in the agent's internal model. Pengi, for example, does not have to model all of the blocks on the video screen. Second, when the planner needs to know if there is an object with a certain relation to a reference object, it does not need to check all visible objects to find out.

These representational advantages have perceptual analogs which are better illustrated by Ulysses because it addresses perception more realistically. First, not only do perceptual routines avoid having to represent all world objects internally, they avoid having to *look* for all the objects. In Pengi perception was abstract and essentially cost-free; for a real driving robot, looking for things is extremely expensive and any reductions in sensing requirements are important. Second, checking the relationship between two objects is more complicated than, say, looking at a value in a property list. Computing arbitrary geometric relations may be difficult, and it is much better to do it once and find the appropriate object directly. For example, finding a car on a road could involve making a region-containment test on every visible car; it would be better to scan the road region until the car was found.

[2]We use the term "perceptual routines" instead of "visual routines" in our work to emphasize that they may include higher levels than just low-level vision, and that they may include other sensing technologies.

4. Driving with Perceptual Routines

4.1. Implementing Ulysses

Execution Architecture. Ulysses has a simple architecture that makes it almost completely reactive. In other words, at each decision time—every 100ms—Ulysses makes a fresh analysis of the situation without using any information from the past (except for a few bits of state). Ulysses thus does not actually make and execute plans, but continuously decides what the robot should do *now*. We are assuming for now that any number of perceptual routines can run in one decision cycle. This scheme allows Ulysses to respond quickly to any new situation.

The reactive system may seem unrealistic because it requires the perception system to do all of its work in a short time interval. However, time can be traded off against the number of cameras and computational resources, so we can still use this model by interpreting cost in terms of *resources* rather than *time*. Future work with Ulysses will investigate architectures that make use of a larger, persistent world model.

Simulated Perception. Most of the cars in PHAROS (which we call "zombies") do not perform any perception or scene interpretation tasks. However, the car controlled by Ulysses gets all of its information about the world through a restricted interface to PHAROS. PHAROS simulates the execution of perceptual routines and passes symbolic information back to Ulysses. By monitoring the interface between the two programs, we can determine exactly what information Ulysses requires at any time. Although the PHAROS world is artificial and devoid of real-world features, we examine perception costs as if the robot were in a more realistic environment.

4.2. Use of Routines at an Intersection

We will now illustrate the use of our perceptual routines in a simulated driving scenario. The situation, shown in Figure 4-1, is that the robot is approaching a left-side road and must turn left. There are a few speed limit signs, a STOP sign, a "left turn only" sign overhead and a "left turn only" arrow on the pavement. The robot's road has four lanes, while the side road has only two.

Since Ulysses is reactive, it uses the perceptual routines over and over again as it drives the robot. Later we will show how the perceptual cost varies during the scenario; here we will show only one

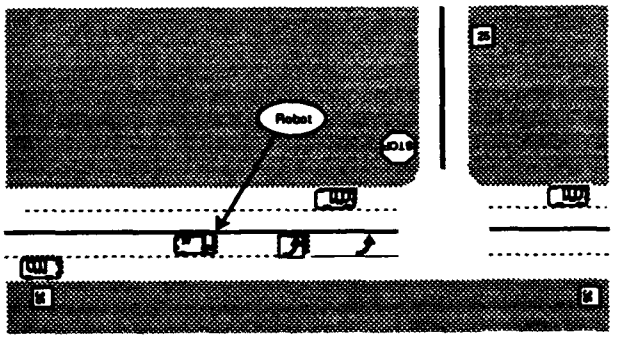


Figure 4-1: Partway through the left side road scenario.

decision cycle. At this point in the scenario there is a car approaching from the through direction, one in the adjacent lane behind, and one in the oncoming lanes (having turned from the side road).

The first thing the robot does is find the lane immediately in front of it. It then tracks the lane forward (stopping at the intersection), looks for cars in the lane, and looks for signs along the side of the road. Figure 4-2 shows the areas scanned by these four routines. The lane tracking routine leaves a marker at the entrance to the intersection.

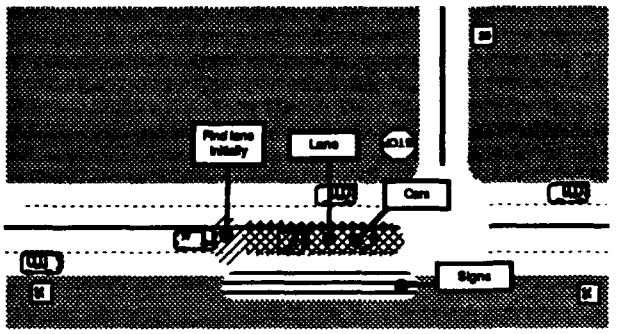


Figure 4-2: The robot scans for the lane, cars, and signs ahead.

Figure 4-3 depicts the process of determining the channelization of the robot's lane (i.e., whether it can turn left from the lane). Ulysses scans the width of the road—profiles it—at the intersection (at the marker left earlier) to determine the position of the robot's lane at the intersection. The robot also looks for signs over the lane and markings in the lane to indicate turn restrictions. These scans are repeated until the routines have reached the end of the lane. Markers indicate where one leaves

off (at an observed sign or marking) and the next one starts.

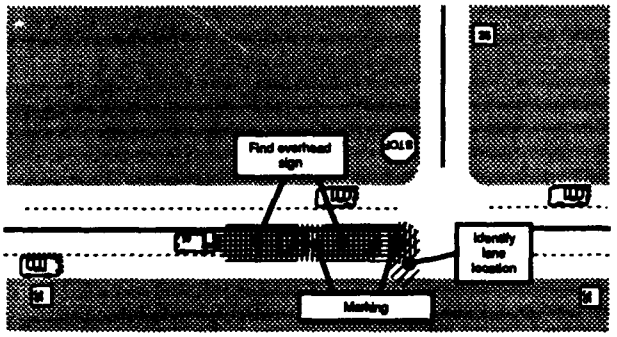


Figure 4-3: The robot checks lane position and looks for channelizing signs and marks.

Next Ulysses analyzes the intersection. This requires searching the intersection area for a path through the intersection, lead cars on this path, crossing or obstructing cars, traffic signals around the intersection, and other approach roads. Information Ulysses already has allows it to determine whether there is a traffic control sign facing the robot, and how many lanes the robot's road has. Lane counts help determine which road is bigger and are part of the right-of-way determination process [14].

Figure 4-4 shows areas scanned to check for approaching traffic on each road, and for STOP or YIELD signs facing this traffic. Ulysses first finds the lanes on the approach roads, and then looks for the closest car to the intersection in each lane.

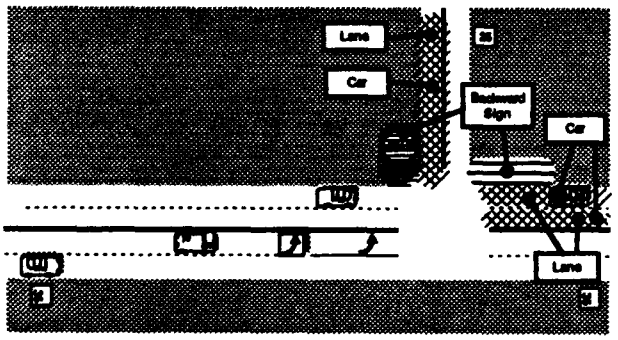


Figure 4-4: The robot checks the intersection approaches for traffic and signs.

In Figure 4-5 the robot is looking beyond the intersection to its downstream road. As with the first lane, Ulysses first finds the lane, then looks for cars and signs. In this case one sign is found.

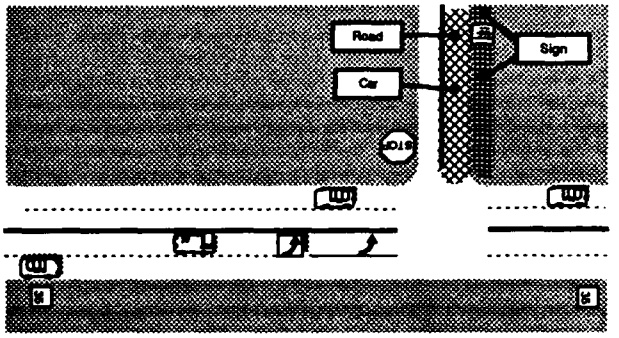


Figure 4-5: The robot looks for a lane, cars, and signs downstream of the intesection.

At this point Ulysses has examined everything necessary to constrain the robot's speed. The remaining steps are performed to select a lane. Ulysses considers lane selection even as the robot approaches the intersection because in some situations it may be desirable or necessary to change lanes to move around a slow car. Figure 4-6 shows that Ulysses finds the adjacent lane (to the right only, in this case) and analyzes it much as the robot's lane. Ulysses tracks the lane ahead to the intersection, and looks for cars, overhead signs (signs to the right have already been found), markings, and lane position at the intersection. The robot also looks in the lane behind to find the car behind.

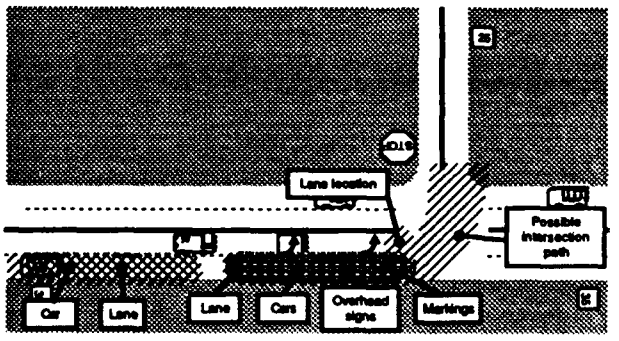


Figure 4-6: The robot looks forward and backward for constraints affecting the adjacent lane, and traffic in that lane.

4.3. The Cost of Using Routines

The perceptual routines are subject to the same environmental conditions as the general perception system, so must pay the same feature extraction costs in general. However, routines can reduce perception costs in several ways.

- **Reduced search area.** The azimuth and elevation angles swept by the routine are much smaller than the area searched by the general perception system.
- **Range-limited resolution.** The maximum depth reached in the routine's search area limits the resolution required.
- **Object-limited resolution.** The routines look for only one type of object at a time; resolution is determined by the size of that type, not by the smallest of all types.
- **Limited features.** Since the routines look for only one type of object at a time, they may be able to use simpler techniques to extract a more limited variety of features from the image.

The effectiveness of these reductions depends on the situation; for example, if the robot used several routines to search the same area for different objects, it would not get the benefit of limited features or object-limited resolution.

In some cases recognition algorithms used by the routines could be simpler than those used in the general system. For example, when looking for a car on a road ahead, it is only necessary to detect a large solid blob. The routine is looking only on a road, so any blob there may reasonably be assumed to be a car. Fewer features are needed to recognize blobs than are needed to recognize cars. However, for the purposes of this paper we assume that the recognition method is the same for routines as it is for the general system.

We calculate the cost of using routines using the same formula we used for the general system (Equation 1). However, we use a feature size particular to the routine, and the particular area searched by the routine (as illustrated by Figures 4-2 through 4-6 above) to calculate the number of pixels. Resolution is also allowed to decrease as the search area gets closer to the robot (as it is in the general perception case). We describe the cost calculations for routines in greater detail in [15].

The perception interface in PHAROS computes the cost of Ulysses' requests automatically. This allows us to experiment with various scenarios and sensing strategies to see how they affect perceptual cost. Figure 4-7 shows the computed cost for the robot during the entire left side-road scenario. The situation illustrated in Figure 4-1 occurs sometime between points (3) and (4). We can see that the perceptual cost of using routines is 3 orders of magnitude less than general perception.

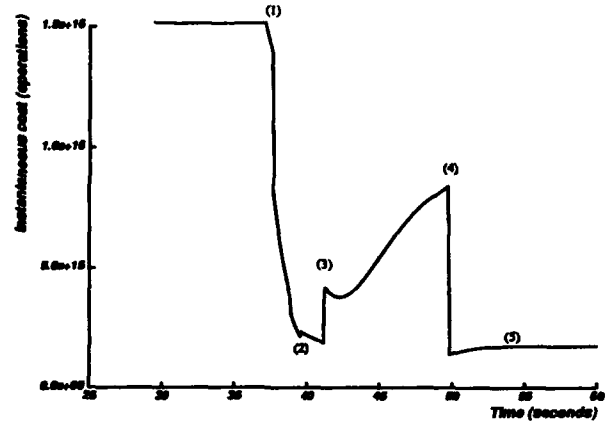


Figure 4-7:

Perceptual cost of using routines during the left side-road scenario. Notes: Before (1) the robot is driving on a 4-lane road; at (1) the robot gets to within sensor range (150m) of the intersection and starts a lane change; at (2) the robot gets to within sensor range of the roads leaving the intersection; at (3) the robot finishes the lane change; at (4) the robot enters the intersection and ceases searching for approaching traffic, traffic control devices, etc.; and at (5) the robot begins driving on the 2-lane road.

5. Conclusions

We have described how tactical driving is a complex task in a complex environment. While clever bottom-up image processing techniques will be necessary for a robot to handle the perceptual load in this domain, they will not be sufficient. A robot must also use task knowledge to constrain the perception problem. We have designed a driving model and used it to control perception for driving—to specify what the robot should look for and where it should look for it. With these constraints, the perception task is simpler and and can be performed more quickly by special purpose routines. A general analysis of the complexity of general perception and simulation of driving using routines shows that routines reduce compuational

cost by several orders of magnitude.

We are continuing to develop our driving model and the perceptual routines for driving. We are also exploring additional ways of reducing perceptual needs, such as using models of the coherence of the world over time. This research will help to discover ways that a robot can perform interesting tasks in perceptually difficult environments.

Acknowledgement

This research was sponsored by the Avionics Lab, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597 and Contract DACA76-85-C-0003.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

References

[1] AASHTO. *A Policy on Geometric Design of Highways and Streets* American Association of State Highway and Transportation Officials, Washington, D.C., 1984.

[2] Agre, P. E. and D. Chapman. Pengi: An Implementation of a Theory of Activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268-272. Morgan Kaufman Publishers, Los Altos, 1987.

[3] Akatsuka, H. and I. Shinichiro. Road Signposts Recognition System. In *Proceedings of the SAE International Congress*. SAE, Detroit, 1987.

[4] Crisman, J. and C. Thorpe. Color Vision for Road Following. In *Proceedings of the SPIE Conference on Mobile Robots*. SPIE, 1988.

[5] Ettinger, G. J. Large Hierarchical Object Recognition Using Libraries of Parametrized Sub-parts. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 32 - 41. IEEE, Ann Arbor, Michigan, June, 1988.

[6] FHWA. *Manual on Uniform Traffic Control Devices* Federal Highway Administration, U.S. Department of Transportation , Washington, D.C., 1978.

[7] Hebert, M. and T. Kanade. 3-D Vision for Outdoor Navigation by an Autonomous Vehicle. In C. Thorpe and T. Kanade (editors), *CMU-RI-TR-88-4: 1987 Year End Report for Road Following at Carnegie Mellon*, pages 29-41. Carnegie Mellon University, 1988.

[8] Kluge, K. and H. Kuga. Car Recognition for the CMU Navlab. In C. Thorpe and T. Kanade (editors), *CMU-RI-TR-88-4: 1987 Year End Report for Road Following at Carnegie Mellon*, pages 99-115. Carnegie Mellon University, 1988.

[9] Legislative Reference Bureau. *Pennsylvania Consolidated Statutes, Title 75: Vehicles (Vehicle Code)* Commonwealth of Pennsylvania, Harrisburg, PA, 1987.

[10] McKeown, D. M. Jr. and J. L. Denlinger. Cooperative Methods For Road Tracking in Aerial Imagery. In *Proceedings of the 1988 DARPA IUS Workshop*, pages 327-341. Morgan Kaufmann, 1988.

[11] McKnight, J. and B. Adams. *Driver Education and Task Analysis Volume I: Task Descriptions*. Final Report, Department of Transportation, National Highway Safety Bureau, Washington, D.C., November, 1970.

[12] Michon, J. A. A Critical View of Driver Behavior Models: What Do We Know, What Should We Do? In L. Evans and R. Schwing (editors), *Human Behavior and Traffic Safety*. Plenum, 1985.

[13] Reece, D. A. and S. Shafer. An Overview of the Pharos Traffic Simulator. In J. A. Rothengatter and R. A. deBruin (editors), *Road User Behaviour: Theory and Practice*. Van Gorcum, Assen, 1988.

[14] Reece, D. and S. Shafer. *A Computational Model of Driving for Autonomous Vehicles*. Technical Report CMU-CS-91-122, Carnegie Mellon University, April, 1991.

[15] Reece, D. and S. Shafer. Using Active Vision to Simplify Perception for Robot Driving. 1991.CMU CS Technical Report, to appear.

[16] Ullman, S. Visual Routines. *Cognition* 18:97-160, 1984.

Combining Artificial Neural Networks and Symbolic Processing for Autonomous Robot Guidance

Dean A. Pomerleau* **Jay Gowdy** **Charles E. Thorpe**
School of Computer Science and Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Artificial neural networks are capable of performing the reactive aspects of autonomous driving, such as staying on the road and avoiding obstacles. This paper describes an efficient technique for training individual networks to perform these reactive driving tasks. But driving requires more than a collection of isolated capabilities. To achieve true autonomy, a system must determine which capabilities should be employed in the current situation to achieve its objectives. Such goal directed behavior is difficult to implement in an entirely connectionist system. This paper describes a rule-based technique for combining multiple artificial neural networks with map-based symbolic reasoning to achieve high level behaviors. The resulting system is not only able to stay on the road, it is able follow a route to a predetermined destination, turning appropriately at intersections and stopping when it has reached its goal.

1 Introduction

Artificial neural networks are commonly employed as monolithic non-linear classifiers. The technique, often used in domains such as speech, character and target recognition, is to train a single network to classify input patterns by showing it many examples from numerous classes. The mapping function from inputs to outputs in these classification tasks can be extremely complex, resulting in slow learning and unintelligible internal representations.

However there is an alternative to this monolithic network approach. By training multiple networks on different aspects of the task, each can learn relatively quickly to become an expert in its sub-domain. This paper describes a technique we have developed to quickly train expert networks for vision-based autonomous vehicle control. Using this technique, specialized networks can be trained in under five minutes to drive in situations such as single-lane road driving, highway driving, and collision avoidance.

Achieving full autonomy requires not only the ability to train individual expert networks, but also the ability to integrate their responses. This paper focuses on rule-based arbitration techniques for combining multiple driving experts into a system that is capable of guiding a vehicle in a variety of circumstances. These techniques are compared with other neural network integration schemes and shown to have a distinct advantage in domains where symbolic knowledge and techniques can be employed in the arbitration process.

2 Driving Module Architecture

The architecture for an individual ALVINN driving module is shown in Figure 1. The input layer consists of a single 30x32 unit "retina" onto which a sensor image from either the video camera or the laser range finder is projected. Each of the 960 input units is fully connected to the hidden layer of 5 units, which is in turn fully connected to the output layer. The 30 unit output layer is a linear representation of the currently appropriate steering direction. The centermost output unit represents the "travel straight ahead" condition, while units to the left and right of center represent successively sharper left and right turns. The steering direction dictated by the network may serve to keep the vehicle on the road or to prevent it from colliding with nearby obstacles, depending on the type of sensor input and the driving situation the network has been trained to handle.

To drive the Navlab, an image from the appropriate sensor is reduced to 30 x 32 pixels and projected onto the input layer. After propagating activation through the network, the output layer's activation profile is translated into a vehicle steering command. The steering direction dictated by the network is taken to be the center of mass of the "hill" of activation surrounding the output unit with the highest activation level. Using the center of mass of activation instead of the most active output unit when determining the direction to steer permits finer steering corrections, thus improving ALVINN's driving accuracy.

3 Individual Driving Module Training and Performance

We have developed a scheme called training "on-the-fly" to quickly teach individual modules to imitate the driving reactions of a person. As a person drives, the network is trained with back-propagation using the latest video camera image as input and the person's current steering direction as the desired output. To facilitate generalization to new situations, additional variety is added to the training exemplars by shifting

*The principle support for the Navlab has come from DARPA, under contracts DACA76-85-C-0019, DACA76-85-C-0003 and DACA76-85-C-0002. This research was also funded in part by a grant from Fujitsu Corporation.

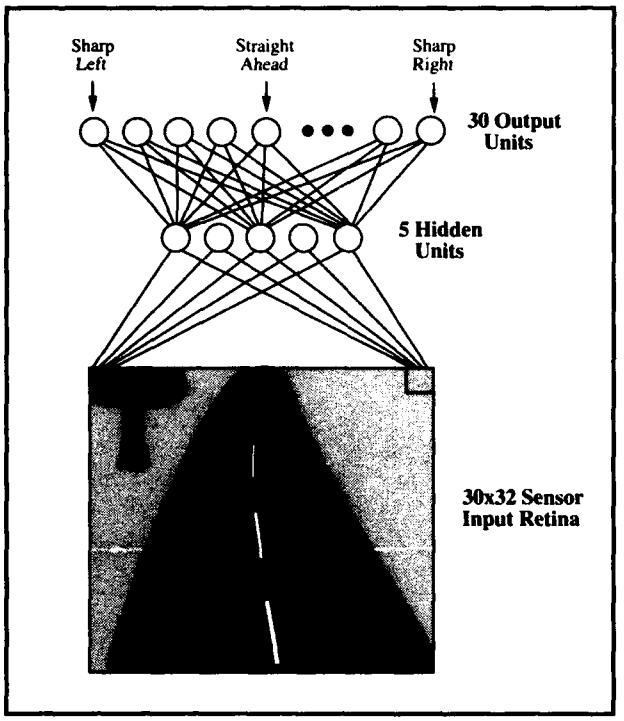


Figure 1: The architecture for an individual ALVINN driving module

and rotating the original camera image in software to make it appear that the vehicle is situated differently relative to the environment (See Figure 2). The correct steering direction as dictated by the driver for the original image is altered for each of the transformed images to account for the altered vehicle placement. Adding these transformed patterns to the training set allows the network to learn to recover from driving mistakes, without requiring the human trainer to explicitly stray from the road center and then return. For more details about the technique and purpose of training on-the-fly, see [Pomerleau, 1991].

Running on three Sun-4 Sparcstations, training on-the-fly requires about five minutes during which a person drives the

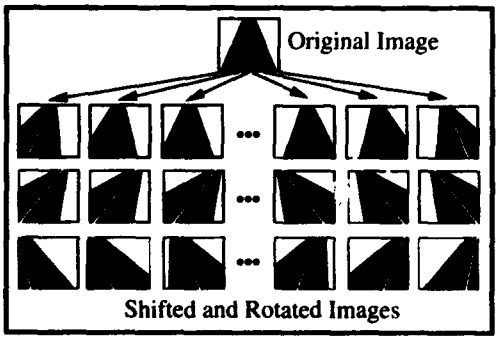


Figure 2: The single original video image is shifted and rotated to create multiple training exemplars in which the vehicle appears to be at a different locations relative to the road.



Figure 3: Video images taken on three of the roads ALVINN modules have been trained to handle. They are, from left to right, a single-lane dirt access road, a single-lane paved bicycle path, and a lined two-lane highway.

Navlab at about six miles per hour over a 1/4 to 1/2 mile stretch of training road. Once it has learned, the network can accurately traverse the length of road used for training and also generalize to drive along parts of the road it has never encountered under a variety of weather conditions. In addition, since determining the steering direction from the input image merely involves a forward sweep through the network, the system is able to process 20 images per second, allowing it to drive at up to the Navlab's maximum speed of 20 miles per hour[1]. This is over twice as fast as any other sensor-based autonomous system has driven the Navlab [Kluge and Thorpe, 1990, Crisman and Thorpe, 1990].

The flexibility provided by training on-the-fly has facilitated the development of individual driving networks to handle numerous situations. Using video camera images as input, networks have been trained to drive on single-lane dirt roads, single-lane paved roads, two-lane suburban neighborhood streets, and lined two-lane highways (See Figure 3).

By replacing the video input with alternative sensor modalities, ALVINN has learned other interesting behaviors. One such sensor onboard the Navlab is a scanning laser range finder. The range finder provides images in which pixel values represent the distance from the range finder to the corresponding area in the scene. Obstacles such as trees and cars appear as discontinuities in depth, as can be seen in the simulated range finder image at the bottom of Figure 4. Using this sensor, separate ALVINN modules have been trained to avoid collisions in obstacle-rich environments and to follow alongside rows of parked cars.

A third type of image used as input to ALVINN modules comes from a laser reflectance sensor. In this type of image, a pixel's value corresponds to the amount of laser light that reflects off the corresponding point in the scene and back to the sensor. The road and off-road regions reflect differently, making them distinguishable in the image (see Figure 4). Laser reflectance images in many ways resemble black and white video images, but have the advantage of being independent of ambient lighting conditions. Using this sensor modality, we have trained a network to follow single-lane roads in total darkness.

4 Symbolic Knowledge and Reasoning

Despite the variety of capabilities exhibited by individual driving networks, until recently the system has been far from

[1]The Navlab has a hydraulic drive system that allows for very precise speed control, but that prevents the vehicle from driving over 20 miles per hour.



Figure 4: Images taken of a scene using the three sensor modalities the system employs as input. From left to right they are a video image, a laser range finder image and a laser reflectance image. Obstacles like trees appear as discontinuities in laser range images. The road and the grass reflect different amounts of laser light, making them distinguishable in laser reflectance images.

truly autonomous. First, the one driving network architecture shown in Figure 1 was capable of driving only on the type of road on which it was trained. If the road characteristics changed, ALVINN would often become confused and stray from the road. In addition, a real autonomous system needs to be capable of planning and traversing a route to a goal. The neural network driving modules are good at reactive tasks such as road following and obstacle avoidance, but the networks have a limited capability for the symbolic tasks necessary for an autonomous mission. The system of networks cannot decide to turn left at an intersection in order to reach a goal. After making a turn from a one lane road to a two lane road, the system does not know that it should stop listening to one network and start listening to another. Just as a human needs symbolic reasoning to guide reactive processes, the networks need a source of symbolic knowledge to plan and execute a mission.

Ideally, the symbolic knowledge source would reason like a person. It would use its knowledge of the world to plan a sequence of observations and corresponding actions to traverse the route. For instance, to achieve the goal of reaching a friend's house, the mission description might be a sequences like, "Drive until the sign for Seneca Road is seen, and turn left at that intersection. Then drive until the third house on the left is seen, and stop in front of it."

In this ideal system, once the mission is planned, the symbolic knowledge source would rely entirely on perception to control the execution of the mission. In other words, the symbolic resource module would be able to recognize events and use what it sees to guide the interaction of the networks. The symbolic resource module would be capable of reading the street sign at an intersection and making the appropriate turn to continue on to its destination. It would also be able to identify the new road type and choose the appropriate network for driving on that kind of road. Unfortunately, the perception capabilities required by such a module are beyond the current state of the art.

In order to bridge the gap between mission requirements and perception capabilities, we use additional geometric and symbolic information stored in an "annotated map". An annotated map is a two dimensional data structure containing geometrical information about the area to be traversed, such as the locations of roads and landmarks. In addition, each object in the map can be annotated with extra information to be interpreted by the clients that access the map. For example, as far as the annotated map is concerned, a mailbox is simply a two dimensional polygon at a particular location with some extra bits associated with it. The "extra bits" might represent the three dimensional shape of the the mailbox, or even the name of the person who owns it. The module which manages the annotated map does not interpret this extra information, but rather provides a mechanism for client modules to access the annotations. This reduces the knowledge bottleneck that can develop in large, completely centralized systems.

The annotated map is not just a passive geometric database, but instead is an active part of our system. Besides having a 2D representation of the physical objects in a region, annotated maps can contain what are called alarms. Alarms are conceptual objects in the map, and can be lines, circles, or regions. Each alarm is annotated with a list of client modules to notify and the information to send to each when the alarm is triggered. When the annotated map manager notices that the vehicle is crossing an alarm on the map, it sends the information to the pertinent modules. Once again, the map manager does not interpret the information: that is up to the client modules.

Alarms can be thought of as positionally based production rules. Instead of using perception based production rules like, "If A is observed, then perform action B", an annotated map based system has rules of the form, "If location A is reached, then perform action B". Thus we reduce the problem of making high level decisions from the difficult task of perceiving and reacting to external events to the relatively simple task of monitoring and updating the vehicle's position.

The first step in building an annotated map is collecting geometric information about the environment. We build our maps by driving the vehicle over roads and linking the road segments together at intersections. At the same time, a laser range finder is used to record the positions of landmarks such as mailboxes and telephone poles. Planning a particular mission requires adding specific instructions to the map in the form of "trigger annotations". This is currently a process performed by the person planning the mission. For example, the human expert knows that when approaching an intersection, the vehicle should slow down, so the expert chooses the appropriate location to put the trigger line. The trigger line goes across the road at that point, and is annotated with a string of bits that represents the new speed of the vehicle. During the run, when the vehicle crosses the trigger line, the map manager sends the string of bits to a module that interprets the information and slows the vehicle to the desired speed. In the current system, alarms are interpreted as commands, but there is no predefined "correct" way for a module to react to an alarm. Depending on its content, an alarm could also be interpreted as a wakeup call, or even as simply advice.

Because position information is so critical to an annotated map system, we use multiple techniques to determine the vehicle's current location. We use an Inertial Navigation System (INS) which can determine the vehicle's location with an error of approximately 1% of distance traveled [Amidi & Thorpe, 1990]. To eliminate positioning error that accumulates over time in the INS data, the annotated map system also uses information from perception modules. For example, since the driving networks presumably keep the vehicle on the road, lateral error in the vehicle positioning system relative to the road can be identified and eliminated. In addition, a module

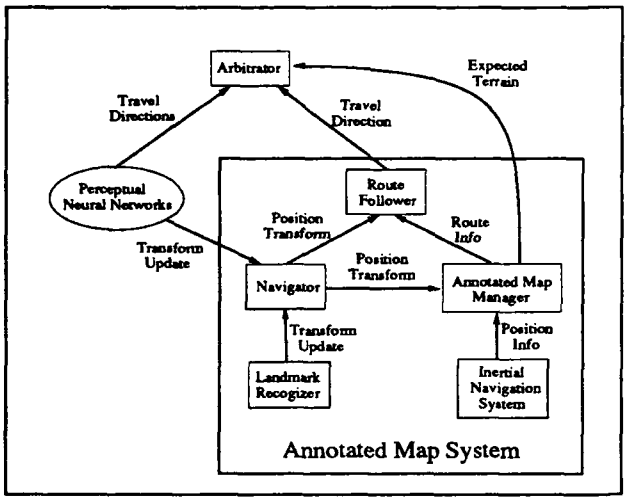


Figure 5: The components of the annotated map system and the interaction between them. The annotated map system keeps track of the vehicle's position on a map. It provides the arbitrator with symbolic information concerning the direction to steer to follow the preplanned route and the terrain the vehicle is currently encountering. The neural network driving modules are condensed for simplicity into a single block labeled perceptual neural networks.

using the laser range finder compares the landmarks it sees to the landmarks collected when the map was built, and triangulates the vehicle's position on the map. These techniques allow perception modules to provide useful positioning information *without* requiring them to explicitly recognize and interpret particular objects such as street signs. The position corrections provided by perception modules are interpreted as a change in the transform between the location that the INS reports and the real vehicle position on the map. A separate module, called the navigator, is in charge of maintaining and distributing this position transform.

Annotated maps provide the system with the symbolic information and control knowledge necessary for a fully autonomous mission. Since the control knowledge is geometrically based, and since planning is done before the mission starts, runtime control comes at a low computational cost. Figure 5 shows the structure and interaction of the annotated map system's components. It also illustrates the annotated map system's interaction with the other parts of the system, including the perceptual neural networks and the arbitrator (discussed below). Figure 6 shows a map and annotations for a mission segment.

5 Rule-based Driving Module Integration

We use the symbolic knowledge provided by the annotated map system to help guide the interaction of the reactive driving neural networks. Figure 7 shows the system architecture with emphasis on the neural networks. Whereas Figure 5 subsumed the neural network systems into one unit labeled "perceptual neural networks", Figure 7 subsumes the annotated map system into one package. In this diagram, each box represents a separate process running in parallel. Images from the three onboard sensors are provided to the five driv-

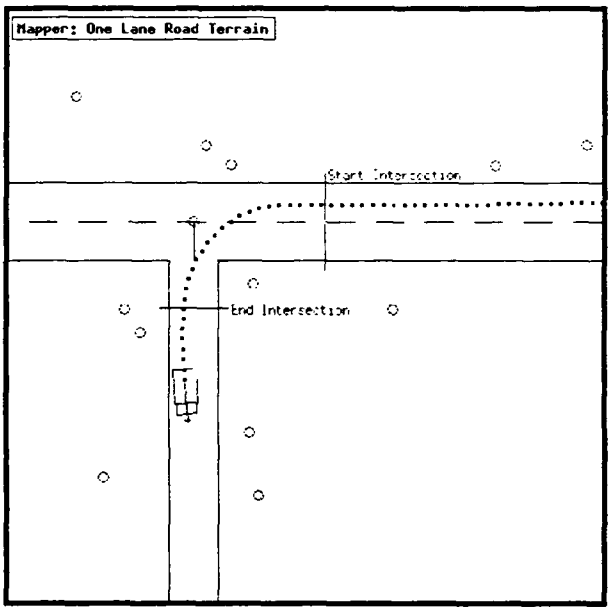


Figure 6: A section of a map created and maintained by the annotated map system. The map shows the vehicle traversing an intersection between a single- and a two-lane road. The lines across the roads are alarms which are triggered when crossed by the vehicle. Triggering an alarm results in a message being passed from the map manager to the arbitrator indicating a change in terrain type. The circles on the map represent the positions of landmarks, such as trees and mailboxes. The annotated map system uses the locations of known landmarks to correct for vehicle positioning errors which accumulate over time.

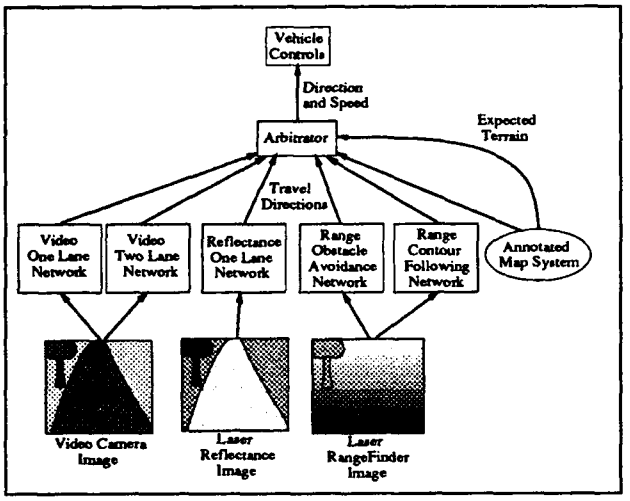


Figure 7: The integrated ALVINN architecture. The arbitrator uses the terrain information provided by the annotated map system as well as symbolic models of the driving networks' capabilities and priorities to determine the appropriate module for controlling the vehicle in the current situation.

ing networks shown in the second row of the diagram. The driving networks propagate activation forward through their weights, with each determining what it considers to be the correct steering direction. These steering directions are sent to the arbitrator, which has the job of deciding which network to attend to and therefore how to steer.

The arbitrator makes use of both the geometric and control information provided by the annotated map system to perform a mission autonomously. First, the route following module within the annotated map system uses the geometric information in the annotated map to recommend a vehicle steering direction. The direction recommended by the route follower is the direction it thinks the vehicle should steer in order to follow the preplanned route. When the vehicle is driving down a road, the route follower queries the annotated map for the position of the road ahead of the vehicle. The route follower uses this geometric information to generate a steering direction.

The annotated map system also provides the arbitrator with information about the current driving situation, including what type of road the vehicle is on, and whether there is an intersection or dangerous permanent obstacle ahead. For example, suppose during the planning phase the human expert notices that at a particular point the road changes from one lane to two. The expert would set a trigger line at the corresponding point on the map and annotate it with a message that will tell the arbitrator to stop listening to the one lane road following network and start listening to the two lane road following network. When the alarm is triggered during the run, the arbitrator combines the advice from the annotated map system with the steering directions of the neural network modules using a technique called relevancy arbitration.

Relevancy arbitration is a straightforward idea. If the annotated map system indicates the vehicle is on a two-lane road, the arbitrator will steer in the direction dictated by the two-lane road driving network, since it is the relevant module for the current situation. If the annotated map system indicates the vehicle is approaching an intersection, the arbitrator will choose to steer in the direction dictated by the annotated map system, since it is the module that knows which way to go in order to head towards the destination. In short, the arbitrator combines symbolic knowledge of driving module capabilities with knowledge of the present terrain to determine the relevant module for the current circumstances.

The relevancy of a module need not be based solely on the current terrain information provided by the annotated map system. Instead, the arbitrator also employs rules for determining a module's relevancy from the content of the module's message. The obstacle avoidance network has one such rule associated with it. The obstacle avoidance network is trained to steer straight when the terrain ahead is clear and to swerve to prevent collisions when confronted with obstacles. The arbitrator gives low relevancy to the obstacle avoidance network when it suggests a straight steering direction, since the arbitrator realizes it is not an applicable knowledge source in this situation. But when it suggests a sharp turn, indicating there is an obstacle in the vehicle's path, the urgency of avoiding a collision takes precedence over other possible actions, and the steering direction is determined by the obstacle avoidance network. This priority arbitration is similar in many ways to the subsumption architecture [Brooks, 1986], although the most common interaction between behaviors in Brooks' systems is for higher level behaviors to override less sophisticated, instinctual ones.

By combining map-related knowledge about the current driving situation with knowledge about abilities and priorities of individual driving modules, the integrated architecture provides the system with capabilities that far exceed those of individual driving modules alone. Using this architecture, the system has successfully followed a 1/2 mile path through a suburban neighborhood from one specific house to another. In navigating the route, the system was required to drive through three intersections onto three different roads while swerving to avoid parked cars along the way. At the end, the vehicle came to rest one meter from its destination.

6 Analysis and Discussion

Rule-based integration of multiple expert networks has significant advantages over previously developed connectionist arbitration schemes. One such advantage is the ease of adding new modules to the system. Using rule-based arbitration, the new module can be trained in isolation to become an expert in a new domain, and then integrated by writing rules for the arbitrator which specify the new module's area of expertise and its priority. This is in contrast to other connectionist expert integration techniques, such as the task decomposition architecture [Jacobs *et. al.*, 1990], connectionist glue [Waibel, 1989] and the meta-pi architecture [Hampshire and Waibel, 1989]. To combine experts using these techniques requires the training of additional neural network structures, either simultaneously with the training of the experts in the case of the task decomposition architecture, or after expert training in the case of the connectionist glue and meta-pi architectures. Adding a new expert using these techniques requires retraining the entire integrating structure from scratch, which involves presenting the system patterns from each of the experts' domains, not just the new one. This large scale retraining is particu-

larly difficult in a task like autonomous navigation because it requires either driving over all the experts' domains again, or storing a large number of domain-specific images for later reuse.

Another significant advantage of rule-based arbitration is the ease with which non-neural network knowledge sources can be integrated into the system. Symbolic tasks such as planning and reasoning about a map are currently difficult to implement using neural networks. In the future, it should be possible to implement more and more symbolic processing using connectionist techniques, but until then, rule-based arbitration provides a way of bridging the gap between neural networks and traditional AI systems.

The technique is not without shortcomings however. The current implementation relies too heavily on the accuracy of the annotated map system, particularly for negotiating intersections. The question might be asked, why is the mapping system required for intersection traversal in the first place? Why can't the driving networks handle intersections? The answer is that when approaching an intersection, an individual driving network will often provide ambiguous steering commands, since there are multiple possible roads to follow. If left on its own, a road-following network will often alternately steer towards one or the other road choices, causing the vehicle to oscillate and eventually drive off the road. In addition, even if the network could learn to definitively choose one of the branches to follow, it still wouldn't know which is the *appropriate* branch to choose in order to head toward the destination. In short, the mapping modules can be viewed both as a useful source of high level symbolic knowledge, and as an interim solution to the difficult perceptual task of intersection navigation.

The annotated map system as currently implemented is not a perfect solution to the problem of high level guidance because it requires both detailed knowledge of the route, and an accurate idea of the current vehicle position. In certain controlled circumstances, such as rural mail delivery, the same route is followed repeatedly, making an accurate map of the domain feasible. However a system capable of following less precise directions, like "go about a half mile and turn left on Seneca Road", is clearly desirable. Such a system would require more reliance on observations from perception modules and less reliance on knowledge of the vehicle's exact position when making high level decisions.

Conceptually, this shift towards reliance on perception for high level guidance could be done in two ways. First, observations of objects like the Seneca Road street sign, could be used to update the vehicle's position on the map. In fact, position updates based on perceptual observations are currently employed by the annotated map system when it triangulates the vehicle's location based on the positions of known landmarks in laser range images. But position updates are only helpful when the observations are location specific. For observations of objects like stop lights, or arbitrarily located objects like "road construction ahead" signs, the system's response should be independent of the vehicle's location.

These location independent observations could be modeled as positionless alarms in the annotated map. When a perception module sees an object like a "road construction ahead" sign, it would notify the map manager. The map manager would treat the sighting as an alarm, distributing the information associated with the alarm to the pertinent modules. Perception triggered alarms would allow the system to transition between its current perceptual abilities and future, more advanced capabilities.

Although the system is not yet capable of identifying and reading individual signs, we have had preliminary success in using neural network perceptual observations to help guide high level reasoning. The technique relies on the fact that when the vehicle reaches an intersection, the output of the driving network becomes ambiguous. This ambiguity manifests itself as an output vector with more than one active steering direction corresponding to the multiple possible branches to follow. This output ambiguity has been successfully employed to update the vehicle's position and to follow coarse directions. As the vehicle approaches an intersection, the annotated map system signals the arbitrator that an intersection is coming up and that the vehicle should follow the right-hand branch in order to head towards the goal. This level of detail does not require either a highly accurate map or precise knowledge of the vehicle's current position. The arbitrator takes the annotated map system's message as a signal to watch the output of the current driving network carefully. When the driving network's output becomes ambiguous, the arbitrator signals the annotated map system that the vehicle has reached the intersection and to update the vehicle's position accordingly. The arbitrator also uses the "turn right" portion of the annotated map system's message in order to choose the correct steering direction from the driving network's ambiguous output vector. This closer interaction between the perception networks and the annotated map allows the system to use perception for intersection traversal, instead of relying solely on knowledge from the map for guidance.

Another shortcoming of rule-based arbitration as currently implemented its binary nature. Currently, a module is deemed by the annotated map system as either appropriate or inappropriate for the current road type. This binary decision does not address the question of intelligently combining modules trained for the same domain, such as the video-based single-lane driving network and the laser reflectance-based single-lane driving network. There are obviously some situations, such as night driving, when one network is better suited than the other. To take more subtle circumstances into account when weighting the steering directions dictated by multiple networks, we are developing augmented arbitration rules that consider more context than just the current road type. We are also currently working on connectionist techniques that can determine a network's reliability directly from its output alone. Preliminary results in this area look very promising.

One final drawback of the current system is the need for a human expert to preplan the mission by providing map annotations. In the future, we will replace the human expert with an expert system capable of annotating the map appropriately. We understand the techniques the human expert uses to find the shortest route and to annotate the map, so automating the process should not be difficult.

In conclusion, a modular architecture permits rapid development of expert neural networks for complex domains like autonomous navigation. Rule-based arbitration is a simple and efficient method for combining these experts when symbolic knowledge is available for reasoning about their appropriateness. Rule-based arbitration also permits the com-

bination of neural network experts with non-neural network processing techniques such as planning, which are difficult to integrate using other arbitration schemes.

References

[Amidi & Thorpe, 1990] O. Amidi and C. Thorpe. Integrated mobile robot control. In *Proceedings SPIE Mobile Robots V, vol. 1388*, pp 504-524, 1990.

[Brooks, 1986] R.A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 1, pp. 14-23, April 1986.

[Crisman and Thorpe, 1990] J.D. Crisman and C. Thorpe. Color Vision for Road Following. In *Vision and Navigation: The CMU Navlab* Charles Thorpe (Ed.), Kluwer Academic Publishers, Boston, 1990.

[Hampshire and Waibel, 1989] J. B. Hampshire and A.H. Waibel. The Meta-Pi Network: Building Distributed Knowledge Representations for Robust Pattern Recognition. Carnegie Mellon Technical Report CMU-CS-89-166-R. August, 1989.

[Jacobs *et. al.*, 1990] R.A. Jacobs, M.I. Jordan and A.G. Barto. Task Decomposition Through Competition in a Modular Connectionist Architecture: The What and Where Vision Tasks. Univ. of Massachusetts Computer and Information Science Technical Report 90-27, March 1990.

[Kluge and Thorpe, 1990] K. Kluge and C. Thorpe. Explicit Models for Robot Road Following. In *Vision and Navigation: The CMU Navlab* Charles Thorpe (Ed.), Kluwer Academic Publishers, Boston, 1990.

[Pomerleau, 1991] D. A. Pomerleau. Efficient Training of Artificial Neural Networks for Autonomous Navigation. In *Neural Computation vol. 3:1*, Terrence Sejnowski (ed)., March, 1991.

[Thorpe and Gowdy, 1990] C. Thorpe, and J. Gowdy. Annotated Maps for Autonomous Land Vehicles. In *Proceedings of the DARPA Image Understanding Workshop*, 1990.

[Waibel, 1989] A. Waibel. Modular Construction of Time Delay Neural Networks for Speech Recognition. In *Neural Computation 1:1*, March, 1989.

Vehicle-Type Motion Modeling and Analysis

Yuncai Liu and Thomas S. Huang

Coordinate Science Laboratory and Beckman Institute
University of Illinois at Urbana-Champaign
405 North Mathews Street, Urbana, IL 61801, U. S. A.

Abstract

A new model for vehicle-type motion is proposed, which assumes that the motion is a rotation around an axis through the vehicle center followed by a forward translation along the main axis of the vehicle. The contribution of this paper is threefold: (1) When the rotation and the amplitude of translation are constant, this type of motion is shown to be equivalent to a constant camera-centered motion. This indicates that a constant motion in the conventional camera-centered model, which is commonly considered artificial, can in fact be a reasonable model in real life. (2) We show that a constant vehicle-type motion can be interpreted as a constant screw motion. (3) A linear algorithm for estimating constant vehicle-type motion is presented and experiments using real scene images are included.

I. Introduction

An important class of rigid object motion is that of vehicles such as cars, ships and aircrafts. We shall call such motion vehicle-type motion. Using conventional camera-centered motion to model vehicle-type motion is not natural for many applications. Resently, several models for three-dimensional (3D) motion have been suggested, among which are those of Shariat and Price [1], and Weng, Huang and Ahuja [2]. In the former, a motion is modeled as that of a rolling wheel. In the latter, a 3D motion is described as a cone rolling without slip around another fixed cone. These models are not particularly suitable for vehicle-type motion estimation.

In this paper, a new model for vehicle-type motion is proposed. This model represents a 3D motion as a rotation around an axis through the vehicle center followed by a forward translation. Several advantages arise with this model: First, this model is much closer to real vehicle motion than existing motion models, especially when the motion is small. Second, in the case of constant motion, for which the rotation and amplitude of translation are assumed invariant, as we shall see later, a vehicle-type motion is equivalent to a constant camera-centered motion. This gives rise to several benefits. (i) constant camera-centered motion, which has been used by many researchers, is commonly thought to be artificial, i. e., few cases of object motion in real life could fit the assumption. The research of this paper, however, discovers that constant camera-centered motion is in fact a good model for vehicle motion. (ii) since a constant vehicle-type motion can be converted into a constant camera-centered motion, it can be estimated by many existing two-view algorithms of motion estimation, e. g., Zhuang, Huang and Haralick [3], Furthermore, we shall show that a constant vehicle motion can be interpreted as a constant screw motion. Finally, for constant vehicle-type motion, a long image sequence can be used to combat noise.

The motivation behind this research is to discover a proper model for vehicle motion estimation using long image sequences. Since a general vehicle motion, if smooth, can be approximated by a piecewise constant motion, the constant vehicle-type motion model has wide applications.

II. Model for Vehicle-type Motion

In most existing techniques of motion estimation, a 3D motion is usually modeled as a camera-centered motion, i. e., the camera is assumed to be fixed at the origin of the world coordinate system with optical axis of the camera coinciding with the z-direction (or negative z-direction), and the motion is represented by a rotation around an axis though the camera center followed by a translation. Assume that $\vec{p}_i$ is a point on a 3D object at time instant t_i; $\vec{p}_{i+1}$ the same point at time instant t_{i+1}, $\mathbf{R}_{oi}$ and $\vec{T}_{oi}$ the rotation and translation, respectively, in this model from time instants t_i to t_{i+1}. The motion is described by

$$\vec{p}_{i+1} = \mathbf{R}_{oi}\vec{p}_i + \vec{T}_{oi} \tag{1}$$

Although it is mathematically convenient, this model appears to lack practical meaning since there are few cases in real life where an object moves in such manner. Especially, in the case of constant motion, the camera-centered model is commonly thought to be artificial.

1. Vehicle-type motion

Suppose there is a vehicle moving in the world coordinate system with the camera center at the origin. Considering a 3D point on a moving vehicle at position $\vec{p}_i$ at time instant t_i, and at position $\vec{p}_{i+1}$ at time instant t_{i+1}, let $\mathbf{R}_i$ be the rotation of the point around an axis through the object center $\vec{c}_i$ of time instant t_i, $\vec{T}_i$ be the translation along the main axis of the object from time instants t_i to t_{i+1}. See Figure 1. Then the motion of this point in the model of vehicle-type motion is described as

$$\vec{p}_{i+1} = \mathbf{R}_i(\vec{p}_i - \vec{c}_i) + \vec{T}_i + \vec{c}_i, \quad i = 0, 1, \ldots N-1 \tag{2}$$

with the constraints

$$\vec{T}_i = k_i \mathbf{R}_i \vec{T}_{i-1}, \quad i = 1, 2, \ldots N-1 \tag{3}$$

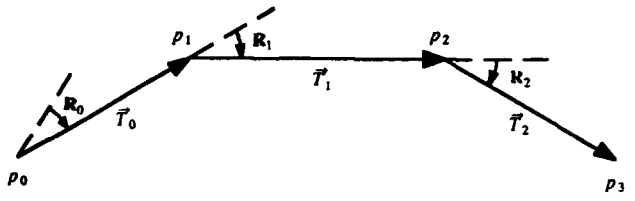


Figure 1. A model of vehicle-type motion

$$\vec{c}_i = \vec{c}_{i-1} + \vec{T}_{i-1}, \quad i = 1, 2, \ldots N \tag{4}$$

where k_i is a scale parameter for translation amplitude. Note that in this model, the center position of an object is determined only by the preceding center position and the translation, but not the rotation.

2. Converting vehicle-type motion to camera-centered motion

In the motion described by Eq. (2), the rotations and the amplitudes of translations for different time instant pairs may have different values. We now consider the special case of constant motion, where the rotations and the amplitudes of translations are assumed constant. This implies that $\mathbf{R}_i = \mathbf{R}$ and $k_i = 1$ for all i, and the formulas for a vehicle motion become

$$\vec{p}_{i+1} = \mathbf{R}(\vec{p}_i - \vec{c}_i) + \vec{T}_i + \vec{c}_i, \quad i = 0, 1, \ldots N-1 \tag{5}$$

$$\vec{T}_i = \mathbf{R}\vec{T}_{i-1}, \quad i = 1, 2, \ldots N-1 \tag{6}$$

$$\vec{c}_i = \vec{c}_{i-1} + \vec{T}_{i-1}, \quad i = 1, 2, \ldots N \tag{7}$$

Substituting Eqs. (6) and (7) into Eq. (5), one obtains

$$\vec{p}_{i+1} = \mathbf{R}(\vec{p}_i - \vec{c}_{i-1} - \vec{T}_{i-1}) + \mathbf{R}\vec{T}_{i-1} + \vec{c}_{i-1} + \vec{T}_{i-1}$$

$$= \mathbf{R}(\vec{p}_i - \vec{c}_{i-1}) + \vec{T}_{i-1} + \vec{c}_{i-1}$$

By recursively substituting, the motion equation can finally be simplied to

$$\vec{p}_{i+1} = \mathbf{R}(\vec{p}_i - \vec{c}_0) + \vec{T}_0 + \vec{c}_0 \tag{8}$$

Noticing that $\vec{T}_0$ and $\vec{c}_0$ remain constant, one can recognize that the motion of Eq. (8) is actually equivalent to that of a camera-centered model. To make it clear, let

$$\vec{Y} = (\mathbf{I} - \mathbf{R})\vec{c}_0 + \vec{T}_0 \tag{9}$$

where $\mathbf{I}$ is a 3×3 unit matrix. Equation (8) can be then written as

$$\vec{p}_{i+1} = \mathbf{R}\vec{p}_i + \vec{Y}, \quad i = 0, 1, \ldots N-1 \tag{10}$$

Equation (10) has the same form as Eq. (1). It describes a motion as a rotation $\mathbf{R}$ around an axis through the camera center followed by a translation $\vec{Y}$. There are many existing techniques that may be used to solve this problem, e. g., Ref [3]. If every image point is visible in all frames, the minimum numbers of points as a function of the number of frames are listed in Table 1.

Table 1. Minimum number of image points as a function of the number of frames

number of frames	minimum number of points
2	8
3	4
4	3
5, 6, 7, 8	2
≥ 9	1

3. Determining motion model parameters

After having calculated $\mathbf{R}$ and $\vec{Y}$, we proceed to determine the motion model parameters in Eq. (5). Since the real object center is very hard to estimate from images, we take as object center the centroid of the 3D points whose images are used for motion estimation. The 3D structure of image points can be estimated from the coordinates of the image points and the motion parameters $\mathbf{R}$ and $\vec{Y}$ found in the preceding section. Using the formulas in Ref. [3], the 3D structure of the image points are

$$\begin{aligned} x &= zX \\ y &= zY \\ z &= -\frac{1}{2}\left[\frac{\| \vec{Y} \times \vec{P}_{j+1} \|}{\| \vec{P}_{j+1} \times \mathbf{R}\vec{P}_j \|} + \frac{\| \vec{Y} \times \mathbf{R}\vec{P}_{j-1} \|}{\| \vec{P}_j \times \mathbf{R}\vec{P}_{j-1} \|}\right] \end{aligned} \tag{11}$$

where $\vec{p}_j = (x, y, z)^T$ is the 3D coordinates of the image point $\vec{P}_j = (X, Y, 1)^T$; $\mathbf{R}$ and $\vec{Y}$ are the rotation matrix and the translation vector, respectively, subject to $\|\vec{Y}\| = 1$, described in the camera centered model. Once the coordinates of the 3D points on the vehicle are estimated, the position of the object center of the vehicle can be computed. Let the object center at time instant i be $\vec{C}_{obi}$. The rotation center of the vehicle is determined by minimizing the distances of the rotation centers and object centers at each instant subject to the constant motion constraints of Eqs. (6) and (7). Let $\vec{c}_i$ be the rotation center of the vehicle at time instant i. then it can be estimated by

$$\sum_{i=0}^{N} \| \vec{c}_i - \vec{C}_{obi} \|^2 = \min, \tag{12}$$

subject to

$$\| \vec{c}_{i+1} - (\mathbf{I} + \mathbf{R})\vec{c}_i + \mathbf{R}\vec{c}_{i-1} \| = 0, \quad i = 1, 2, \ldots, N-1$$

After the rotation centers $\vec{c}_i$ are computed, the translation vectors of the object centers at each time instant are easily determined.

$$\vec{T}_0 = \vec{Y} + (\mathbf{R} - I)\vec{c}_0 \tag{13}$$

$$\vec{T}_i = \mathbf{R}\vec{T}_{i-1}, \qquad i = 1,2, \ldots N-1 \tag{14}$$

Thus, the motion is completely described by the rotation $\mathbf{R}$, the rotation centers $\vec{c}_i$ and translation vectors $\vec{T}_i$ in the model of object centered-motion.

III. Interpreting Vehicle-Type Motion by Screw Decomposition

A 3D motion may has various interpretations depending on the constraints on the descriptions of rotation or translation, e. g., the choice of a rotation center. In this section, we show that a constant vehicle-type motion can be interpreted as a constant screw motion if the rotation centers at all time instants are constrained to lie on a straight line called the screw axis. The model of a screw motion has intuitive appeal. In particular, the vehicle-type motion of a ground vehicle can be considered as motion along a circle around a fixed point. The parameters describing a screw motion are the screw axis, the rotation radius, the rotation angle and the translation vector along the screw axis.

If the screw axis is written in parameter form

$$\vec{p} = \hat{n}t + \vec{C}_0 \tag{15}$$

where $\vec{p}$ is a general 3D point on the screw axis; $\hat{n}$ is the direction of the screw axis determined from $\mathbf{R}$; t is a parameter; $\vec{C}_0$ is a point the screw axis passes, which can be computed by

$$\begin{cases} (\mathbf{I} - \mathbf{R})\,\vec{C}_0 = \vec{Y} - (\vec{Y} \cdot \hat{n})\hat{n} \\ z = 0 \end{cases} \tag{16}$$

Once the rotation axis is found, the screw radius, which is defined as the radius of the screw cylinder, can be estimated relatively to within a scale factor, just like the translations.

$$r = \frac{1}{N} \sum_{i=0}^{N-1} \| \det[\hat{U} \mid \hat{n} \mid \vec{C}_0 - \vec{C}_{obi}]^T \| \tag{17}$$

where $\hat{U} = [\hat{x}, \hat{y}, \hat{z}]^T$, and $\hat{x}$, $\hat{y}$ and $\hat{z}$ are the unit vectors in x-, y-, and z-directions of the 3D coordinate system. Also, the rotation angle between any two adjacent frames can be easily computed from the rotation matrix.

IV. Experiment Results

The images used in the experiment were of an outdoor scene containing a moving ground vehicle in five image frames (L13 - L17). Several target points had been marked on the vehicle, four of them were visible in those specific frames. The 2D coordinates of these target points on the image plane were used as input data for motion estimation. The details of the image data can be found in Ref. [3].

The world coordinates used in Ref. [3] is a right-handed system fixed on the ground with its x-axis and z-axis on the horizontal ground plane and y-axis vertically upward. The camera is mounted on the ground such that the optical axis of the camera is in the negative z-direction of the ground coordinate system. The camera coordinate system, which has its origin at the camera center with the x- and y-axes parallel to those of the image coordinates, z-axis to the negative optical axis, relates to



Figure 2. A typical frame of intensity image

the world coordinate system by a rotation matrix **M** [3] followed by a translation. The camera used to take the images is a fixed focus A.M.I / Bronica SQ-AM metric 700 mm camera with a lens of 40 mm nominal focal length. The image size is 55.6×55.6 mm^2. The target point positions on the image planes are measured by a one micrometer mono-comparator. The coordinates of these target point can be found in Appendix B of Ref. [3].

The image target points obtained by the mono-comparator have to be processed before they can be used for motion estimation. The major stages of the processing include: (1) lens distortion correction, (2) camera alignment. In the experiment, the positions and orientations of the camera are accurately measured, and then used to align the camera for By the linear algorithm of Zhuang et al, the results of motion estimation for the image of Figure 2 are listed in Tables 2-6. Table

Table 2. Estimated motion using camera-centered model

rotation axis			rotation angle	translation		
n_1	n_2	n_3	degree	x	y	z
-0.32676	0.94163	0.08098	7.06906	0.97218	0.07776	-0.22095

Table 3. The motion parameters measured in camera-centered ground coordinate system

rotation axis			rotation angle	translation		
n_1	n_2	n_3	degree	x	y	z
-0.32768	0.93642	0.12548	7.06906	0.92369	0.09288	-0.37171

Table 4. Structural object centers at each time instant

time instant	x	y	z
13	-2.38055	-0.16021	-9.89579
14	-2.55930	-0.17120	-9.76508
15	-2.72131	-0.18121	-9.61566
16	-2.86464	-0.19004	-9.44998
17	-2.98762	-0.19749	-9.27062

Table 5. Estimated motion using object-centered model

time interval	rotation axis			rotation angle	displacement vector		
	n_1	n_2	n_3	degree	x	y	z
13 - 14	-0.3277	0.9364	0.1255	7.0691	-0.1788	-0.0110	0.1307
14 - 15	-0.3277	0.9364	0.1255	7.0691	-0.1620	-0.0100	0.1494
15 - 16	-0.3277	0.9364	0.1255	7.0691	-0.1433	-0.0088	0.1657
16 - 17	-0.3277	0.9364	0.1255	7.0691	-0.1230	-0.0074	0.1794

Table 6. Estimated motion described as screw motion

screw axis $\vec{p} = \vec{n} + \vec{C}_0$	$\vec{n}$	n_1	n_2	n_3
		-0.32768	0.93642	0.12548
	$\vec{C}_0$	x	y	z
		-1.45794	-0.13349	-7.68458
radius	2.25946			
rotation angle (in degree)	7.066906			
translation along screw axis	0.26234			

Table 7. The computed and the measured distances between target points (measured in mm)

time instant	computed target distance			measured target distance		
	21 - 20	22 - 21	20 - 22	21 - 20	22 - 21	20 - 21
13	3144.46	4327.80	7037.50	3178.18	4251.32	7080.25
14	3139.51	4336.86	7033.38	3178.18	4251.32	7080.25
15	3165.39	4320.08	7024.28	3178.18	4251.32	7080.25
16	3175.53	4322.74	7011.48	3178.18	4251.32	7080.25
17	3143.69	4366.39	6999.68	3178.18	4251.32	7080.25

7 gives a comparison between computed and measured distances between target points. The motion trajectories of the vehicle for the three different motion models are also shown in Figure 3.

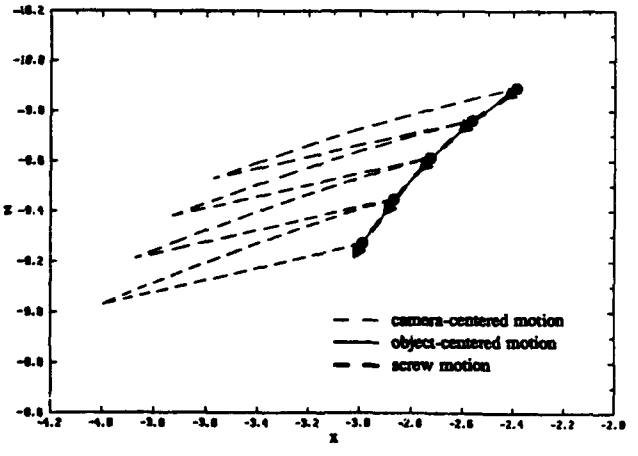


Figure 3. Motion trajectories in three different models

V. Conclusion

A model for vehicle-type motion is proposed in this paper, which may be applied to the motion of many man-made vehicles. Under the constant motion assumption, we show that:

(1) A constant camera-centered motion is equivalent to a constant vehicle-type motion. In the past, constant camera-centered motion is commonly considered as fictitious. This paper has shown that constant camera-centered motion can be a good model for the motion of vehicles.

(2) A constant vehicle-type motion is a constant screw motion. This gives vehicle motion a very vivid interpretation. When a moving object is a ground vehicle, the motion trajectory is a circle.

(3) If a vehicle motion is constant or almost constant, this model makes it possible to use long image sequences for motion estimation and thus make the algorithm robust. A linear algorithm based on existing two-view methods is represented, and experimental results on a carefully calibrated image sequence are included.

Although this paper describes mainly the case of constant motion, the model proposed may be used to approximate a long sequence of general vehicle motion piecewise if the motion is smooth. We are currently investigating this, as well as more complicated special cases of vehicle-type motion.

References

[1] H. Shariat and K. Price, "Motion estimation with more than two frames," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-12, no. 5, pp. 417-434, May 1990.

[2] J. Weng, T. S. Huang, and N. Ahuja, "3D motion estimation, understanding, and prediction from noisy image sequences," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, no. 3, pp. 370-389, May 1987.

[3] E. M. Mikhail and F. C. Paderes, "Photogrammetric series of moving vehicle," Science Service Program DAAL03-86-0001(0863), CAI-RI, U. S. Army ETL, Fort Belvoir, Virginia 22060-5546, Nov. 1988.

Acknowledgement

Authors would like to thank the National Science Foundation for the Grant IRI-89-02728 and the State of Illinois Department of Commerce and Community Affairs for the Grant SCCA 90-82140.

Proximity Detection Using a Filter Tuned in Three-Space

Ian Horswill
MIT Artificial Intelligence Lab
545 Technology Square
Cambridge, MA 02139
ian@ai.mit.edu

Abstract

In this paper I present a simple stereo-based proximity detector and derive asymptotic bounds on its sensitivity using a Bayesian analysis of its performance. The system can be considered to be a filter tuned to a region of three-space. It is extremely fast, running in real time on a personal computer, and is easily parallelized. Thus it is an efficient alternative to deriving a full depth map when proximity information is needed. Since most of its operations must be performed by any edge-based stereo system, it can be used in conjunction with more complicated, higher latency systems at little or no additional cost.[1]

1 Introduction

The construction of fully-general visual systems has proven very difficult. Even simple modules for computations such as stereo matching or structure-from-motion have proven very difficult to make robust and efficient. Recently, there have been a number of calls for examining qualitative information [10][1] or for basing vision research on concrete tasks [5][1][2][6]. One approach is to examine simple activities which agents must commonly perform such as navigation and manipulation. By examining these activities we can determine what information is needed in order to perform them. We can then look at the problem of extracting these different pieces of information as well-defined computational problems.

In this paper, I will examine a particular type of information, proximity, which is useful for tasks such as navigation, and present a simple stereo-based proximity detector.

Proximity detection is the problem of determining whether there is an object within a given distance from the agent. The most obvious application of proximity detection is collision avoidance. What is important is what proximity detection does not involve: it does not require reporting the exact distance or shape of the object, or how many objects, or what their boundaries are—it is a one-bit answer. This reduced information requirement allows the use of dramatically simpler machinery[2].

Formally, we'll define proximity detection to be the problem of determining whether there is an object within some region of 3-space, R, called the *sensitive region* of the detector. Here R is defined relative to, and moves with, the agent's body. The system presented in this paper limits R to a one-parameter family of regions for a given camera configuration and choice of resolution (see below).

2 Algorithm description

A typical edge-based stereo system extracts edges from both images, finds possible matches between edges in the left and right images and then uses some sort of optimization procedure to resolve conflicts between possible matches, so as to determine for each edge point its disparity and hence its depth [3]. For proximity detection, we are concerned only with determining when an object comes within a certain distance of the viewer. Thus it would be sufficient simply to determine whether any points had disparities corresponding to the correct depth range. A further simplification is to note only when objects come into range. This requires only determining if there are any edge points at the disparity corresponding to the outer boundary of the sensitive region of the proximity detector.

[1]Support for this research was provided in part by the University Research Initiative under Office of Naval Research contract N00014-86-K-0685, and in part by the Advanced Research Projects Agency under Office of Naval Research contract N00014-85-K-0124.

[2]For another example of a simplified stereo system see [9].

2.1 A simple algorithm

If we knew for some reason that each edge point had an unambiguous match, we could omit the optimization step mentioned above. Since we are only concerned with a particular disparity d, we would only have to match at that one disparity. This could be accomplished with the following procedure:

```
match-frames(left, right, d)
  le = find-edges(left)
  lr = find-edges(right)
  matches = 0
  for each row i and column j
     if le(i,j) and lr(i,j+d) are
         compatible edge points
     then increment matches
  return matches
```

where by "compatible edges points" I mean edge points which, for example, have the same sign. If the algorithm returns a non-zero value, then there must be an object at the specified disparity.

The disadvantage of the algorithm is that matches between frames are rarely unambiguous or perfect. However, this algorithm has the advantages that (1) it is very simple to implement, (2) it does not require high resolution to help disambiguate matches, (3) it is very fast and can therefore detect collisions with very low latency, (4) it is easy to parallelize on mesh-type hardware, and (5) the basic tasks of edge detection, shifting and matching at a given disparity are likely to be required for *any* edge-based stereo algorithm, thus the algorithm could easily be run in parallel with a more complicated, higher latency algorithm at effectively no cost[3].

The algorithm can be considered to be a spatial filter, but a filter tuned in three-space, not in the normal image-plane spatial domain. It is sensitive to a region of bounded disparity whose thickness is determined by the quantisation of disparities. In particular, it is the set of points in 3-space which project to points in the images with disparity in the interval $(d-\frac{1}{2}, d+\frac{1}{2})$.

2.2 How bad is the simple algorithm?

Let's assume that our edge detector is good so that we can ignore matching errors due to noise in our edge detector. If we place an object in the sensitive region of the filter we will get some response r which is due to three components: a noise term n which is the total number of false matches when the object is not present in the sensitive region (i.e. the response of the filter to the background), a *cross-section* term c which is the total number of matched edge points on the object, and a final term o which is the number of false-positive matches which are occluded by the object, that is, which would have contributed to the response of the filter if the object were not present. Thus total response r is

$$r = n + c - o$$

If we neglect the o term, then we are left with a simple signal which we would like to estimate, corrupted with additive, but not necessarily gaussian, noise:

$$r = n + c$$

Suppose we want to make a simple threshold-based detector for the filter which answers that there is an object present in the sensitive region iff r is greater than some threshold T. We can scale the variables r, n, and c to the interval $[0,1]$ so that they each represent a percentage of matched pixels, and treat r, c and n as random variables. If we assume that c and n are independent with probability densities C and N respectively[4], then we can determine the *a priori* probabilities of false positives and false negatives for various settings of the threshold T.

Let α be the *a priori* probability of there being an object in the sensitive region. For a false positive the reading is entirely due to noise. Since the conditional probability density function of r given that there is no object in the sensitive region is simply N, and the *a priori* probability of there being no object in the sensitive region is $1-\alpha$, we have that the *a priori* probability P_{FP} of a false positive with threshold T is

$$P_{FP} = (1-\alpha)\int_T^1 N(r)dr \qquad (1)$$

Since c and n are independent, we have that the probability of getting a response r_0 given that there is an object in the sensitive region is

$$P(r = r_0|\text{positive}) = \int_0^{r_0} C(x)N(r_0 - x)dx \qquad (2)$$

Thus the *a priori* probability P_{FN} of a false negative given threshold T is

$$P_{FN} = \alpha\int_0^T \int_0^r C(x)N(r-x)dxdr \qquad (3)$$

Beyond this, nothing can be said without assuming something about the forms of C and N. If we assume that the noise is bounded by some *noise level* n_0, that is that $N(x) = 0$ for $x > n_0$, then we can

[3] For example, it could be run in conjunction with the Marr-Poggio 1975 algorithm [7] simply by tapping one layer of the disparity-selective cells and summing their outputs.

[4] This is for explanatory purposes only. Such distributions are not normally known.

always unambiguously detect objects with cross sections greater than n_0. If we are only concerned with detecting objects with large cross sections then the simple algorithm with a threshold $T > n_0$ will suffice. Alternatively, if N has a long tail, we can still treat the noise as being bounded by some arbitrary n_0 at the price of a false positive rate of

$$(1-\alpha)\int_{n_0}^{1} N(x)dx$$

while retaining the property that we will always detect objects with cross sections greater than n_0.

2.3 Noise and sensitivity

Not surprisingly, the noise level limits the size of the objects which we can detect in the sensitive region. Although the precise limit is in terms of the cross section of the object, with a few assumptions we can translate this into a bound on object volume which will give us the order of growth of the worst case sensitivity with noise. Let's define the *worst case threshold volume* to be the least volume V_T such that any object with volume at least V_T will be unambiguously detectable.

Since we are concerned with how small an object we can detect, we'll concern ourselves with objects which fit entirely within the sensitive region[5] It is also necessary to distinguish between what I will call *well-textured* objects, which have sufficient texture to cause the edge detector to fire all over their surface, and *untextured* objects which only reliably produce a response from the edge detector along their generating contours (image-plane boundaries).

Let P be the projection of the object. The volume V of the object is bounded above by the product of the thickness of R, the maximum depth depth of R, the focal length of the lens, and the area of P. All but the latter are constants, thus $V = O(P)$. This is sufficient to give us the order of growth of the threshold volume with noise:

Theorem 1 *For small objects (objects which fit in R), the order of growth of the worst case threshold volume of the simple detector is $O(n_0)$ for well textured objects or $O(n_0^2)$ for untextured objects.*

Proof: Recall that the detector can unambiguously detect any object with cross section greater than n_0. Since $P = \Omega(V)$, we need only show that the cross section is $\Omega(P)$ for textured objects or $\Omega(P^{1/2})$ for untextured objects respectively[6]. In the textured case, we have that the cross section is bounded below by some constant times the area of P, thus $c = \Omega(P)$. For untextured objects, we must at least be able to sense the generating contour of the object. The region with the least boundary per unit area is a circle, for which the area is quadratic in the length of the boundary. Thus $c = \Omega(P^{1/2})$.

3 Strategies for coping with noise

In practice, there is always matching noise. I will discuss three possible modifications to the algorithm to reduce the noise level.

3.1 Ignoring noise

Before complicating the algorithm, it is worth evaluating just what the impact of noise is on the system. As was mentioned above, the simple version of the algorithm will suffice if objects tend to have large cross sections (i.e. are large and textured).

Some environments will produce very little noise. For example, if there is very little texture in the environment and it is widely spaced, then noise is less likely to be a problem. Unfortunately, this is very difficult to quantify. Empirically, we can always ignore the noise, set a threshold, and see what the results are. Analytically however, the best we can do is assume an unrealistic noise model and use it to characterize the response of the algorithm.

Let's ignore depth effects and model scene texture as a binary random field in which each pixel independently assumes a value of 1 (edge pixel) with probability β, or 0 (no edge) with probability $1-\beta$. This is roughly like viewing a random-dot display at infinity. Then in order for a pixel in the background to be falsely matched, both it and the pixel to which the matching is occurring must be marked as edge points. Since they are independent, this occurs with probability β^2, and so the expected fraction of pixels which will be falsely matched will also be β^2. Thus $n_0 = O(\beta^2)$.

Given this, we can consider the noise level to be quadratic in the density of texture in the scene.

Theorem 2 *For small objects in front of a binary random field, the order of growth of the worst-case threshold volume of the simple detector is $O(\beta^2)$ for well-textured objects, or $O(\beta^4)$ for untextured objects.*

Proof: Follows from theorem 1 and the above analysis.

[5] The analysis of larger objects is highly shape-dependent. There are a number of pathological cases which can break the detector such as very long cones with little texture coming head-on.

[6] Ω is roughly the opposite of O. To say that $f(n) = \Omega(g(n))$ means that for sufficiently large n, f is bounded *below* by some constant times g.

Of course binary random fields at infinity are not a realistic model of the average scene. The result is useful however, both because it is reasonable to expect the quadratic dependence of the noise level on the level of texture in the background to carry through to other domains, and because it points out how unreasonable it is to expect the simple detector to find untextured objects in a cluttered background.

3.2 Filtering

Since disparity decreases with increasing distance[7], we need only insure that any edges in the background are sufficiently far apart that they cannot be confused with each other at the disparity we're concerned with. This can be accomplished with band-pass filtering as with the Marr-Poggio-Grimson stereo algorithm[4] or simply with low-pass filtering (smoothing). When an image is band-pass filtered, its zero-crossings occur on average with the frequency of the filter's center-frequency. Without actually band-pass filtering, we can still reduce the noise level by low-pass filtering. This is the same as moving to a coarser point in scale space.

Unfortunately filtering introduces its own distortions. The filtering can disturb the locations of the edges, thus causing false-negatives. In addition, filtering limits the size of the texture, thus limiting the sensitivity of the detector.

3.3 Improved edge matching

It is also possible to reduce noise by making the similarity measure used for local matching of edges more selective. For example, orientation, slope or intensity might be taken into account, or a wider area of support might be matched. I have not experimented with this approach.

3.4 Accommodation

Another tactic is to directly modify the optical properties of the imaging system itself. Since the transfer function of a lens varies with depth, adding increasing filtering as we move farther from the plane of fixation, we can automatically blur texture far from the sensitive region by focusing the lens on the sensitive region. Pentland has successfully used blurring to estimate depth in synthetic images [8]. We can also converge the cameras on the sensitive region so that it is at zero disparity. This technique has been used both in the Marr-Poggio-Grimson stereo algorithm [7][4] and by the animate vision system of the Rochester robot [2]. These techniques have the advantage that they are depth dependent—they have their maximum effect on points far from the sensitive region. Thus even false matches are much more likely to be due to texture near the sensitive region.

[7]For roughly parallel cameras.

4 Experimentation

I have implemented a version of the simple algorithm and performed preliminary experiments with live data in an office environment. The implemented system samples the image at a resolution of 64×48 pixels. Two cameras with 3mm fixed-focus lenses (110 degree field of view) and a baseline of 65mm are used. The optic axes of the cameras were roughly parallel but were not precisely aligned. The implementation first smoothes the images with a 3×3 kernel and then marks pixels which are strong local maxima of the x derivative of intensity (vertical edges) with the sign of the derivative. Finally, edges are matched for sign. None of the improvements listed in section 3 were used.

The system is implemented on a Macintosh IIx using a home-brew lisp compiler. The current implementation is I/O bound, the time to process a frame being only 142msec, while the time to grab the two frames and display debugging information is at least 200ms, depending on the amount of debugging information. A port to a high-performance digital signal processor is underway. The source code used in the tests is available from the author. Sample results for scenes with distant and far objects are shown in figure 2.

Test results for six objects (three people, a chair, a fire extinguisher, and a toy cow) are given in figure 1. The figure gives *depth-tuning curves*, the graphs of response vs. depth, for each object. The tuning curve is a useful measure of performance. The data show that the detector is tuned to a depth interval of approximately two to four feet[8]. Another useful performance measure is peak signal-to-noise ratio, also given in the figure. The worst S/N ratio was 3dB, meaning that the peak response to the object was still twice the background noise level. This is a fairly comfortable margin.

The system had the easiest time detecting people since people are relatively large and well textured. The fire extinguisher and toy cow were more difficult because they were smaller and poorly textured. The toy cow was an extreme case, being only 7 inches tall and occupying only 60-70 pixels when at the center of the sensitive region. The limited dynamic range of the cameras was a problem. Intensity values of lightly colored surfaces were generally saturated, significantly perturbing the response of the edge detector near saturated regions while entirely removing

[8]The author apologizes for performing the experiments in feet, rather than meters as is done in *real* science, but he couldn't find a metric tape measure.

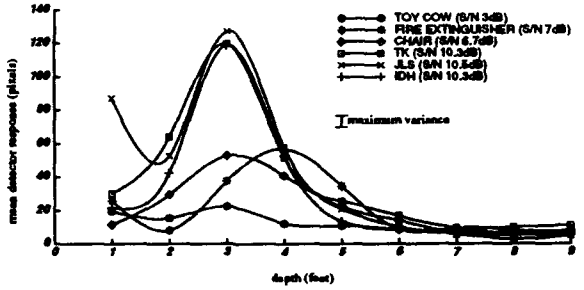


Figure 1: Depth-tuning curves and peak S/N ratios for various objects. S/N ratios for objects are measured as the ratio of peak response to that object over the noise level (12 pixels). Images are 64 × 48 pixels, cameras have 110 degree fields of view with a baseline of 65mm. Objects IDH, JLS, and TK are people. Each object was tested in nine positions of varying depth for 20 trials each. The largest variance was 6.9 pixels. Readings past 7 feet are entirely matching noise. The noise level was measured as the largest such noise value observed over all trials for all data points. The background was a cluttered office.

edges within those regions. Moreover, this saturation varied between the two cameras so that disparities near saturated areas were largely meaningless. The curved, specular surface of the fire extinguisher was also difficult for the detector.

These experiments were limited by the inability to mount the stereo pair on a mobile platform. I hope to use the system for navigation on a mobile robot later this fall.

5 Conclusion

The system presented here is simple, robust, fast, easily parallelizable, and can be implemented in combination with more complicated stereo systems for little or no computational cost. This is not to imply that it is suitable for all tasks. Quite the contrary. The system is an example of how knowledge of a task can simplify the information required from a vision module and thus dramatically simplify the computational machinery needed to implement it.

This raises several questions. First, how broad a range of tasks can be supported by such simple systems? Second, is there a small set of simple systems which together can perform a broad range of tasks? Finally, how do we learn from and generalize such simple, domain-specific systems? The first two are empirical, the latter more philosophical. A common complaint about domain-specific processing is that it is "just a bunch of hacks" which don't teach us anything about the real problems. I believe that these systems can be analyzed in ways which allow us both to make predictions about performance and to apply what we've learned to other problems. With the analysis in this paper, I hope convince the reader that we can learn more from these systems than just the fact that they work.

Acknowledgments

This paper benefited greatly from discussions with Rod Brooks, David Michael, and Lynn Stein. Tom Knight and Jeanne Speckman provided help with the experiments.

References

[1] John Aloimonos. Purposive and qualitative active vision. In *DARPA Image Understanding Workshop*, 1990.

[2] Dana H. Ballard. Animate vision. *Artificial Intelligence*, 48(1):57–86, 1991.

[3] Stephen T. Barnard and Martin A. Fischler. Computational and biological models of stereo vision. In *Proc. DARPA Image Understanding Workshop*, September 1990.

[4] W. E. L. Grimson. Computational experiments with a feature-based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(1):17–34, January 1985.

[5] Ian D. Horswill. Reactive navigation for mobile robots. Master's thesis, Massachusetts Institute of Technology, June 1988.

[6] Katsushi Ikeuchi and Martial Herbert. Task oriented vision. In *DARPA Image Understanding Workshop*, 1990.

[7] David Marr. *Vision.* W. H. Freeman and Co., 1982.

[8] Alex P. Pentland. A new sense for depth of field. In *Proc. Int. Joint Conf. Artificial Intelligence*, pages 988–994, August 1985.

[9] K. Storjohann, T. Zeilke, H. A. Mallot, and W. von Seelen. Visual obstacle detection for automatically guided vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 761–766, May 1990.

[10] W.B. Thompson and J.K. Kearney. Inexact vision. In *Workshop on Motion: Representation and Analysis*, 1986.

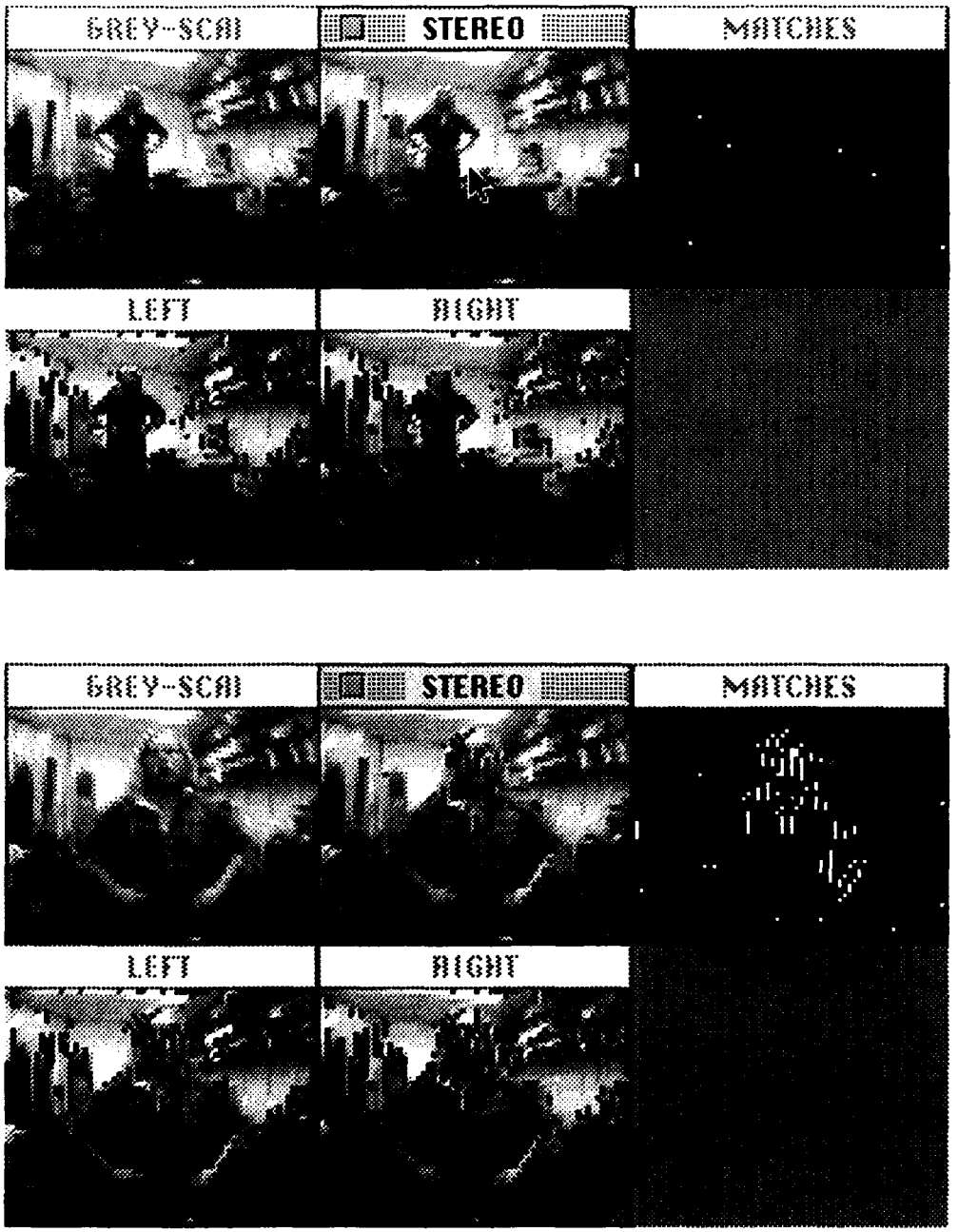


Figure 2: Performance of the system in an unmodified office environment. "Grey-scale" is the original sampled image from the left camera, "left" and "right" are the respective images overlaid with derived edges, "matches" is the set of edges matched between images at disparity 2, and "stereo" is the left grey scale image overlaid with matched edges.

Stealth Terrain Navigation With Bounding Overwatch

Y. Ansel Teng Daniel DeMenthon Larry S. Davis*
Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3411

Abstract

In this paper, we extend our stealth terrain navigation approach to plan paths such that two groups of vehicles move in a bounding overwatch manner. Furthermore, the planned paths for the vehicles themselves are subject to intervisibility constraints, configuration constraints, and different terrain traversabilities due to the variations in terrain type and slope. A spatial-temporal sampling approach is adopted to discretize the solution space and facilitate fast computation on a massively parallel machine. One of the key computations in the planning is region-to-region visibility analysis, for which a fast parallel algorithm is described. The algorithms are implemented on a Connection Machine CM-2, and the experimental results show that the planning system effectively generates good paths.

1 Introduction

This paper describes a path planning method for stealth terrain navigation with bounding overwatch using a massively parallel machine. This is an extension of the stealth terrain navigation approach used in [7]. We consider the problem of planning paths for two groups of vehicles, each of which consists of two vehicles, from their common initial location to their common final goal. The terrain through which they must move is "hostile" in the sense that there are adversaries moving through the terrain. The vehicles should remain hidden from the adversaries to the greatest extent possible. Initial information is available concerning the locations and movements of these adversaries, but it is expected that these models will degrade over time, so that the plan developed must support reconnaissance activities. This requirement should be fulfilled in a bounding overwatch manner such that one of the two groups serves as observer while the other group moves along a safe path to a new observation point using the information collected from the observer; the two groups then switch roles. Furthermore, the planned paths for the vehicles themselves are subject to intervisibility constraints for line of sight communication, and configuration constraints such that the two vehicles in a group move in parallel, and the progress of the vehicles through the terrain is differentially impeded by terrain type and slope. Generally, these problems are instances of path planning in two dimensional space with time varying constraints. Such problems are known to be computationally hard [2, 5]. This will lead us to the development of heuristic and approximate algorithms that can avoid a direct assault on these combinatorial problems, while at the same time developing demonstrably good solutions to such problems.

Our basic approach is to represent the problem using discretizations of space and time, and to develop massively parallel algorithms for the fundamental underlying computations (e.g, visibility analysis, reachability on terrain). The discretization allows many basic computations to be arranged in a regular pattern, and therefore solved in parallel efficiently.

The remainder of this paper is organized as follows: In Section 2 we describe our planning scheme and its application to the above problem. Section 3 describes the visibility analysis algorithms which are essential to the choice of path points for maximal safety and observation points for the reconnaissance activities. The algorithms are implemented on a Connection Machine CM-2 [4] and experimental results are given in Section 4.

2 The path planning scheme

In our path planning scheme, the path planning process is divided into stages, and the two groups take turns serving as an observer while a path is planned for the other group at each stage. The planning process in one stage is referred to as a *subplanning process*, and the length of a stage, referred to as the *subplan interval*, is usually determined before the subplanning process. Since the entire planning process is done by repeating the subplanning process, the following discussion focuses on the subplanning process. Since a spatial sampling approach is adopted, the terrain is represented by a regular grid with elevation data at each grid cell.

We first describe the subplanning process for a single agent, and then extend the approach to plan for a group of agents. The criteria for choosing the subgoal in the bounding overwatch problem are described at the end of

*The support of the Defense Advanced Research Projects Agency (ARPA Order No. 6350) and the U.S. Army Engineer Topographic Laboratories under Contract DACA76-88-C-0008 is gratefully acknowledged.

this section.

2.1 Subplanning for a Single Agent

In the subplan for a single agent, a subgoal is chosen from the area that is reachable within the subplan interval by a good path. In the case of a single agent, a "good" path usually refers to a path that remains hidden from the adversaries as much as possible by taking advantage of the terrain. This property is referred to as *safety*.

Since it is very difficult to represent the visibility map of a moving object analytically, a temporal-sampling approach is adopted, in which a sampling period is determined in advance, a visibility map is computed for each sampling period, and the safety is defined by these samples. The visibility map consists of a binary value on each grid cell and represents the regions visible from the predicted adversary locations. It is computed by the point-to-region visibility analysis algorithm [1, 7], which will also be described in Section 3.2.

These visibility maps are combined with terrain traversability using a dynamic programming paradigm to compute the reachable region and evaluate the safety of the best path to each grid cell in the region. This computation is performed incrementally for each sampling period throughout the subplan interval. The data structure can be described as a path credit table

$$Path\_credit(P, t)$$

where P is the cell index and t is the time index. Each element counts the number of safe points along the safest path from the initial location to the cell P at time t. Initially,

$$Path\_credit(P, 0) = \begin{cases} 1 & \text{if cell } P \text{ is the initial location} \\ 0 & \text{otherwise} \end{cases}$$

The table is computed slice-wise for each sampling period from $t = 0$ to the end of the subplan interval, and the computation for each element depends on the terrain traversability.

In our method, terrain traversability is modeled by discretizing the directions in which the agent can move through a grid cell and associating a traversal cost for each of them as the amount of time needed to pass through the cell in that given direction. The cost can be determined by the slope, the type of terrain, and other factors that may affect the mobility. Using this modeling of terrain traversability, the *Path_credit* table is computed according to the following recurrence relation:

$$Path\_credit(P, t) =$$
$$Max\{\ Path\_credit(P, t-1) + Safety(P, t),$$
$$Path\_credit(Q,\ t - Tcost(Q, P))$$
$$+ Safety\_count(Q,\ t - Tcost(Q, P) + 1,\ t)\ \}$$

where Q is any neighbor of P

$Tcost(Q, P)$ = the traversal cost through Q in the direction to P

$$Safety(P, t) = \begin{cases} 1 & \text{if cell } P \text{ is safe at time } t \\ 0 & \text{otherwise} \end{cases}$$

$$Safety\_count(P, t_1, t_2) = \sum_{t=t_1}^{t_2} Safety(P, t)$$

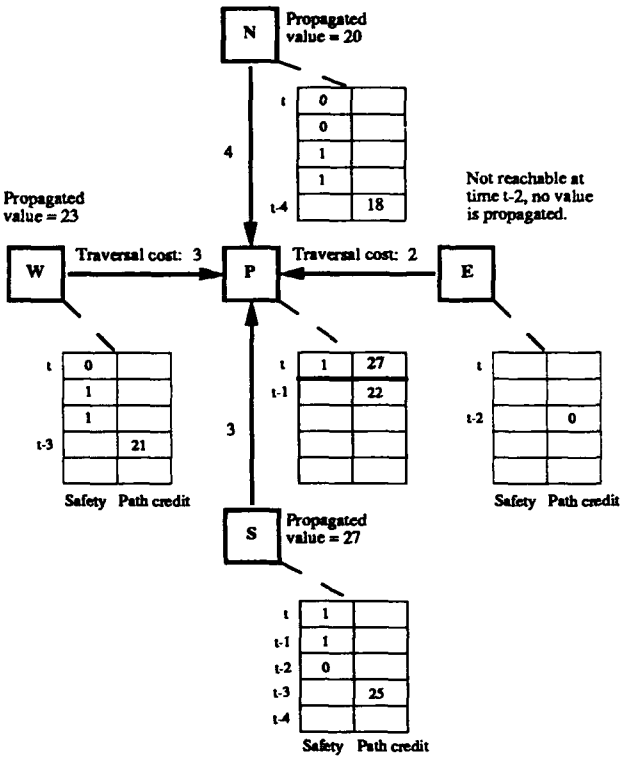


Figure 1: Construction of the path credit table with consideration of terrain traversability. For each neighbor of P, the propagated value is the sum of all values shown in the two-column table associated with each cell.

See Figure 1 for an illustration of this formula, in which four directions are considered for traversing through a cell. One way to interpret this formula is that each grid cell propagates delayed path credits to its neighbors. Each path credit is delayed by an amount of time equal to the traversal cost before it is propagated to the corresponding neighbor and it is updated with safety information during the delayed period. After receiving the path credits from its neighbors, each cell decides its new path credit by choosing the maximum among the received values and its own previous path credit which is also updated by adding the current safety to it.

Clearly, $Path\_credit(P, t) > 0$ if and only if cell P can be reached from the initial location at time t. Thus, the subgoal is chosen only from cells with path credit greater than zero at the end of the interval. After the subgoal is chosen, the path can be extracted from the path credit table by a gradient-following method, or more efficiently, by storing a pointer for each element to the previous cell along the best path during the construction of the table and tracing these pointers back to the initial location.

The following algorithm summarizes the subplanning process for a single agent:

Algorithm: subplanning process for a single agent
begin
for every terrain grid cell P
begin
if P is the initial location of the agent **then**
$Path\_credit(P, 0) = 1$;
else
$Path\_credit(P, 0) = 0$;
end if
for $t = 0$ **to** $end\_of\_subplan\_interval$ **do**
begin
update $Path\_credit(P, t)$ by the recurrence formula;
$Prev(P, t)$ = the pointer to the cell that contributes to the value of $Path\_credit(P, t)$;
end for
end for every
choose a subgoal;
$Path(end\_of\_subplan\_interval)$ = the subgoal position;
for $t = end\_of\_subplan\_interval$ **to** 1 **do**
$Path(t-1) = Prev(Path(t), t)$;
end Algorithm

2.2 Subplanning for a Group of Agents

Now we extend the method to plan a path for a group of agents so that they can maintain a given configuration and optimize their safety and mutual visibility during their movement. In our problem, the agents in a group are required to maintain their configuration as a line segment perpendicular to their direction of motion. A line segment can be represented by the location of its center point and its orientation. With the additional time-axis, the dimensionality of the solution space is four. Our framework can be applied directly in this four-dimensional search space, but it is more efficient to make a further reduction of the dimensionality by applying a decomposition.

First we represent the locations of the agents by the center of the segment, and find a good path for the center, where "good" means a high likelihood that a segment centered at this cell will be safe and intervisible, no matter what orientation the segment is in. This path is actually a corridor for the segment, and the agents are allocated within the corridor using some simple heuristics. Since the corridor has a statistically good evaluation for all the constraints, we should be able to find an acceptable allocation easily. Therefore, we can apply the subplanning method for a single agent to the subplanning for the group center by propagating a path credit defined on both safety and intervisibility.

For a cell to be a potential group center, safety is modeled as the number of safe cells within a circle centered at the cell with a diameter equal to the segment length. This information has to be computed for each temporal sample. For efficiency on regular grids, the circle is approximated by the circumscribing square.

Intervisibility is measured by sampling several line segments centered at a grid cell, and counting the number of pairs of cells that can see each other along these line segments. It needs to be computed only once due to its invariance with time. Currently we sample the horizontal, vertical, and the two diagonal segments to take advantage of the local communication links of the regular grids so that the computation can be done on the terrain efficiently. The intervisibility on each line segment is computed by pipelining the sequential version of the line-visibility algorithm [7, 1].

A weighted sum of safety and intervisibility determines the *point credit* of a cell to be the group center. A threshold on point credit determines the goodness of a cell as a path point, and the path credit is defined as the number of good path points along the best path. The path credit table is constructed the same way as in the case for a single agent. After the path is found, the orientations of the segments are determined by first setting them to the ones perpendicular to the direction of movement, and then smoothing them to avoid large changes between consecutive samples.

2.3 Choosing the Subgoal

The choice of a subgoal depends upon the specific mission in each problem instance. In the bounding overwatch case, the following criteria are considered:

1. Reachability: it must be reachable at the end of the subplan interval;
2. Configuration: its location should be ahead of the current observer by an adequate distance and within a corridor predetermined for the entire movement so that the bounding overwatch pattern can be maintained;
3. Path quality: it should be reachable by a path that is good in terms of safety and intervisibility;
4. Future safety: it should be safe for the next subplan interval;
5. Observability: it should have good observation points in its vicinity for monitoring the movement of the adversaries in the next stage.

The reachability and configuration criteria are satisfied by considering only cells with path credit greater than zero and ahead of the observers by about half of the maximum distance the agents can travel in a subplan interval. Path quality is evaluated by the path credit table. The future safety and observability criteria require computation of the visibility from these candidate cells to the predicted trajectories of the adversaries in the next stage. As the model of the adversary movements degrades with time, the possible trajectories of the adversaries usually span a fairly large region. Since the visibility computation is expensive and these candidate cells usually cluster in regions, a region-to-region visibility analysis algorithm was developed [6] to achieve much faster computation than applying the point-to-region visibility analysis to each candidate cell. This algorithm is briefly described in the next section. After these computations, the subgoal is chosen from the candidate cells by combining all the criteria using a weighted sum and/or thresholds.

3 Algorithms for Visibility Analysis

In this section, we describe the algorithms we used for the three visibility analyses in our planning system: the

line-visibility analysis, the point-to-region visibility analysis (PRVA), and the region-to-region visibility analysis (RRVA). All algorithms are based on digital terrain models and are designed for massively parallel hypercube machines. Hypercube machines provide the flexibility to embed a mesh-connected array of any dimension and the efficiency to perform parallel prefix operations along any axis of the array in logarithmic time. These advantages are used extensively in these algorithms; these kinds of parallel prefix operations will be referred to as *scan* operations.

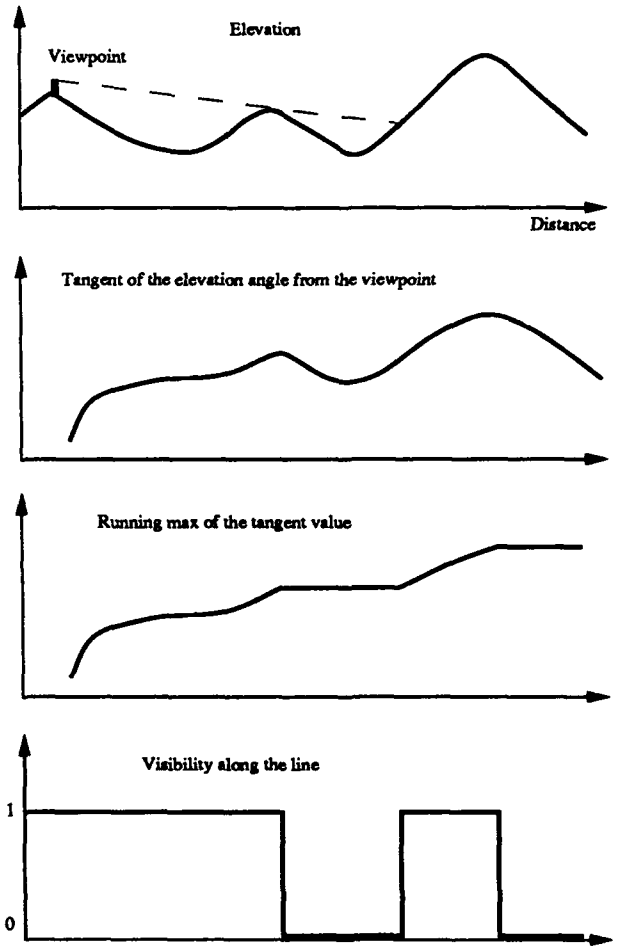


Figure 2: Computing visibility along a line. The X-axis represents the horizontal distance from the viewpoint. The effect of the middle hill is shown by the dashed line in the elevation plot.

3.1 Visibility along a line

Visibility between two points is defined by drawing a line between the two points. The two points are visible to each other if and only if the line lies completely above the terrain. The basic parallel algorithm for visibility analysis is to compute the visibility w.r.t. a given viewing point for every grid cell along a line on the plane projection of the terrain. Suppose we have a list of processing elements (PEs) representing this line. Each of them contains the elevation of the corresponding grid cell on the line with the first cell as the viewing point. The visibility of each grid cell from the viewing point can be determined by first computing the elevation angle from the viewing point to each cell and then comparing the angle with the maximal angle among all cells closer to the viewing point. If its angle is greater than the maximal angle among the cells before it, then this cell is visible. The steps are illustrated in Figure 2. The total complexity is dominated by the computation of the maximal angle before each cell, which can be done by a scan operation in $O(\log W)$ time on a hypercube machine using W processors, where W is the number of grid cells along the line.

3.2 Point-to-region visibility analysis

The PRVA algorithm [1] computes the visibility from a viewing point to a region. Since the underlying terrain is represented by an array of processors, each corresponding to a grid cell of the terrain, the result is returned as a *visibility map* by indicating the visibility of each grid cell with a flag in the associated PE. For computational efficiency, the region is assumed to be an upright rectangle. For a region of arbitrary shape, we may obtain the maximal and minimal X and Y coordinates of the region and use the circumscribing upright rectangle.

We use the term *ray* to refer to a line segment with a direction on the plane projection of a terrain geographically, while it refers to a list of PEs computationally. The processor structure representing a set of rays is called a *ray structure*. We define a *far side* as a side of the rectangle such that when drawing a line from the viewing point to any non-end point on that side, the line will pass through the interior of the rectangle. The minimal set of lines covering all grid cells in a rectangular box will be the lines from the viewing point to all grid points on the far sides. Therefore the PRVA can be done by constructing a ray structure for each far side of the rectangle and running the line-visibility algorithm for each ray in the ray structures.

Let L be the length of a side of the rectangle, and W be the maximal number of grid cells on a single ray to this far side. An $L \times W$ 2-D array of processors is allocated for the ray structure, as shown in Figure 3. Each row in the ray structure corresponds to a ray. After broadcasting the coordinates of the viewing point and the end points of the far side, each processor can find its corresponding grid cell by the digital differential analyzer (DDA) technique [3], and thus obtain the elevation data. The line-visibility algorithm is then conducted along all rays simultaneously. The result will be sent back to its corresponding grid cell. If several results are sent back to the same grid cell, they are combined by an OR operation.

The complexity of this parallel algorithm is

$$O(LW \log W + LW \times Comm) \text{ operations}$$

with minimal time

$$O(\log W + Comm)$$

using $L \times W$ processors, where $Comm$ stands for the complexity of a global communication.

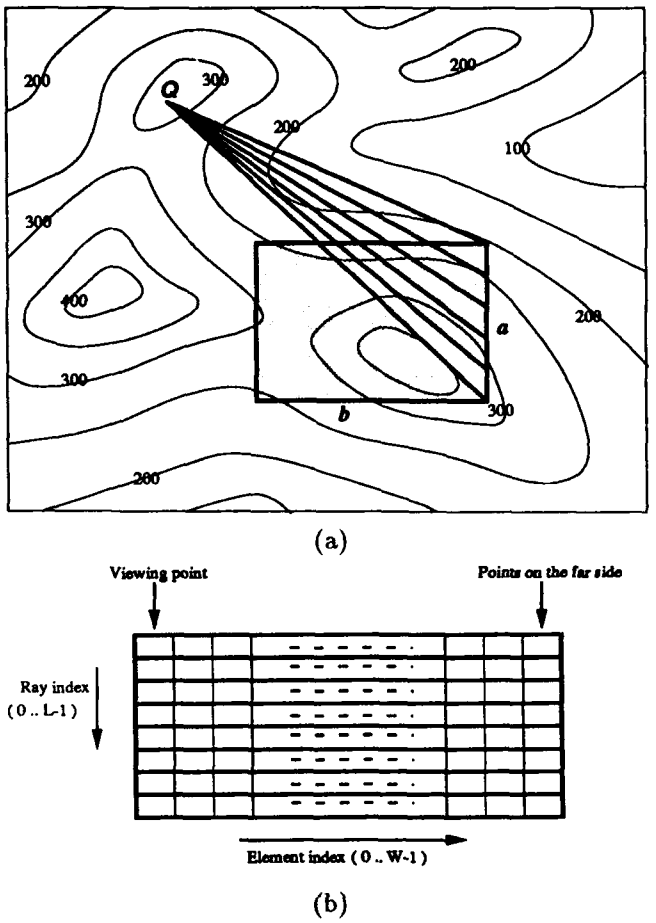


Figure 3: Ray and ray structure. (a) The rays are shown by line segments between the viewing point V and side a of the rectangle. The ray structure consists of all rays for this side and is mapped to a triangle on the terrain map. Note that only sides a and b are far sides. (b) A 2D array is allocated as the ray structure.

As we can see from Figure 3, many PEs in different rays may be mapped to the same terrain cell. Thus concurrent reads/writes occur in the global communication between the two processor structures. An improvement can be made to eliminate the concurrent reads/writes. It can be shown that all PEs that are mapped to the same terrain cell have the same element index and consecutive ray indices. Using this coherence property, the PEs can be grouped by the terrain cell they are mapped to, and only one from each group will participate in the global communication. In each group, the data can be distributed to or combined from every member by segmented scan operations. As this operation is conducted along the dimension of size L, and we expect L to be $O(W)$, it will not increase the complexity.

3.3 Region-to-region visibility analysis

The region-to-region visibility analysis (RRVA) problem is defined as follows: given a source region S and a destination region D on the terrain, compute a measure of visibility from region S to region D. In terms of discretized geometry, this problem can be restated as computing for every point in S the number of visible points in D. In the RRVA algorithm, both the source and the destination regions are assumed to be upright rectangles, but they can be in any relationship, e.g., they can be overlapping or even identical. From the previous subsection, there evidently exists a brute-force solution by applying the PRVA to each grid cell in the source region. The complexity of this brute-force algorithm is $O(L_S^2 L_D W \log W + L_S^2 L_D W \times Comm)$ operations, using up to $L_S^2 L_D W$ processors, where L_S and L_D are the linear size of the source and the destination respectively. Judging from the usual problem size, no existing machines can provide this number of processors. Even if such a machine existed, the number of concurrent reads would make this algorithm virtually impractical. Therefore, we have to stage the analysis in several iterations, each of which computes a subproblem of the entire analysis. Since the communication patterns among iterations will be redundant, we identify some important coherence properties and explain how they can be used to improve the efficiency of communication.

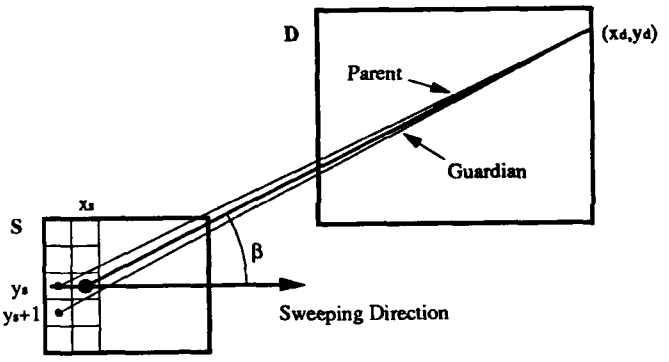


Figure 4: An observation on the coherence property.

One important coherence property is depicted in Figure 4, which shows that the ray from the source point (x_s, y_s) to the destination point (x_d, y_d) is enclosed by the triangle formed by the rays from $(x_s - 1, y_s)$ and $(x_s - 1, y_s + 1)$ to the same destination point.[1] The two rays are referred to as the *parent* and the *guardian* respectively, as shown in the figure. Furthermore, the vertical width of the triangle formed by the parent and the guardian is always less than 1 from x_s to x_d. If the visibility analysis is conducted for a column of the source at a time, and *sweeps* from left to right using the same ray structure, then the PEs in this ray should be able to obtain the elevation data of the corresponding terrain cells from one of the two rays for the previous column using only local communications within the ray structure.

This observation suggests the idea of a sweeping al-

[1] For a better mapping between the geometry in the figures and the ordering in the data structure, a screen coordinate system is adopted in all figures, i.e., the origin is in the upper left corner and the Y-coordinates increase downward.

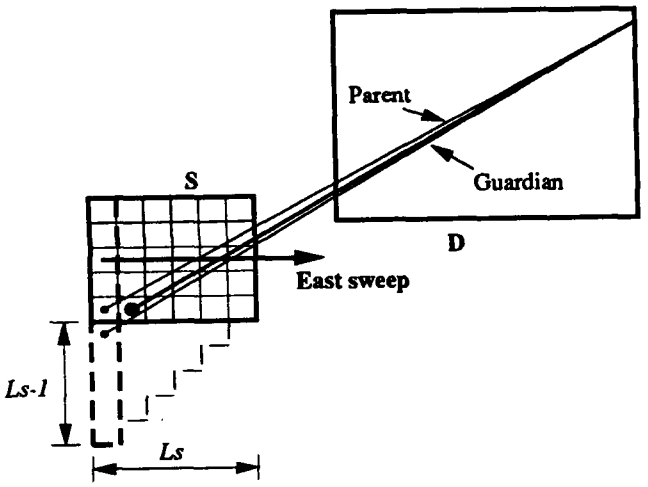


Figure 5: Extended source: the thick dashed line shows the extended initial strip, while the thin dashed lines show how the strip shrinks during the sweep.

gorithm, which sweeps across the source horizontally and/or vertically. Since the above property holds only when the *emitting angle*, β in the figure, is no more than 45 degrees, a partition is defined to divide the rays into four sweeps, namely the East, West, North, and South sweeps, so that the emitting angle is always no more than 45 degrees. In each sweep, the analysis is conducted strip by strip, which is a row or a column of cells that is perpendicular to the sweeping direction. The elevation data are obtained by global communications only for the first strip of each sweep and are passed within the ray structure for the subsequent strips. It can be proved that a ray, its parent, and its guardian belong to the same sweep if they exist. In order to ensure the existence of the guardian for all rays, it may be necessary to extend the source region by the side length along the sweeping direction minus one, and to start the sweep with the *extended initial strip*. Figure 5 shows one such situation. Using these properties, the algorithm is outlined as follows:

```
Algorithm: RRVA
  begin
    for all grid cells in the source region
        set visible-count to 0;
    for each sweep if it is necessary
        construct the 3D ray structure;
        for all PEs in the ray structure
            compute the corresponding grid cell on the
                terrain for the initial strip;
            get the elevation data from the terrain map
                for the initial strip;
            for each strip
                compute the visibility along each ray;
                combine the result in the ray structure;
                update the ray structure for the next
                    strip;
            end for each strip
        send the result back to the source cells on the
            terrain;
    end for each sweep
  end Algorithm
```

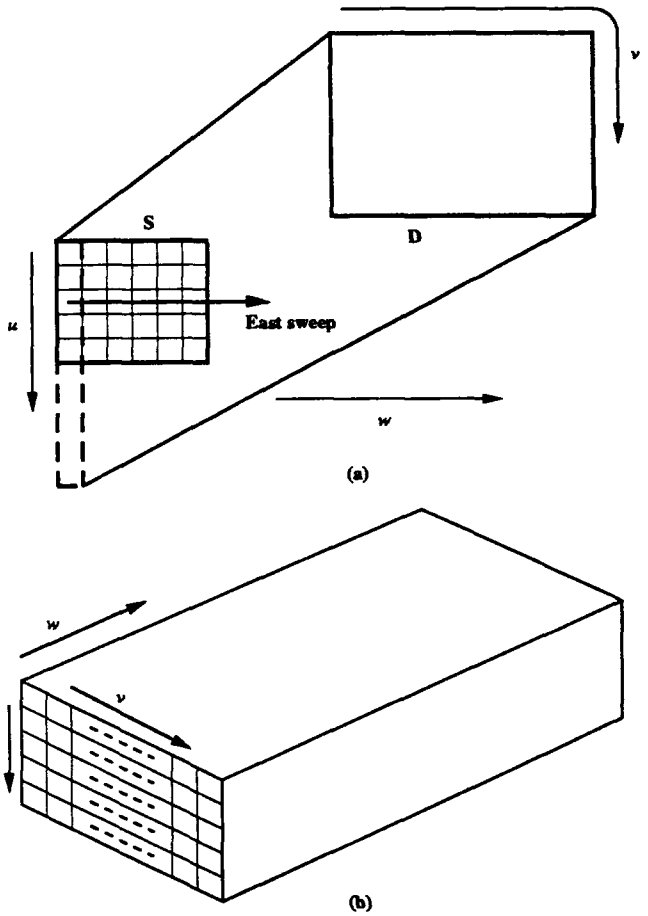


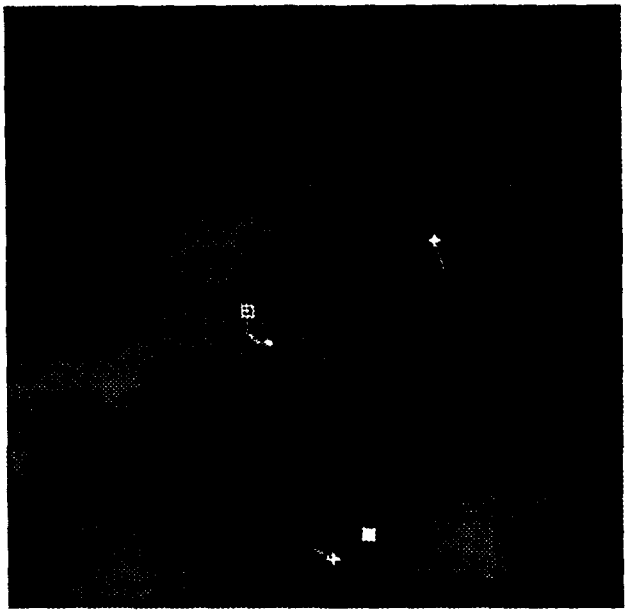
Figure 6: The ray structure for RRVA: (a) the relevant elements on the terrain map; (b) a 3D array of processors is allocated as the ray structure.

Visible-count is the memory location in each terrain cell that returns the number of visible destination cells. The algorithm consists of three major parts:

1. Construction of the ray structure

   The ray structure contains all rays from each source cell on the (possibly extended) initial strip to every grid cell along the far sides of the destination that belongs to the sweep. Since a ray actually contains a list of processors, the entire ray structure is a 3D array of processors, which is indexed by three indices, as shown in Figure 6: u is the index for the grid cells along the strip in the source region, v is the index for the grid cells along the far sides of the destination region, and w is the index of grid cells along the ray.

   When the value of one index is fixed, the other two indices specify a 2D slice of the structure. For example, a *v-w slice* refers to the 2D array for a fixed value of u. It corresponds to all rays starting at the same source point and is actually a part of the 2D ray structure for the PRVA of that source point. We also use the terms *u-neighbors*, *v-neighbors*, and *w-neighbors* to specify the neighboring elements along each axis.

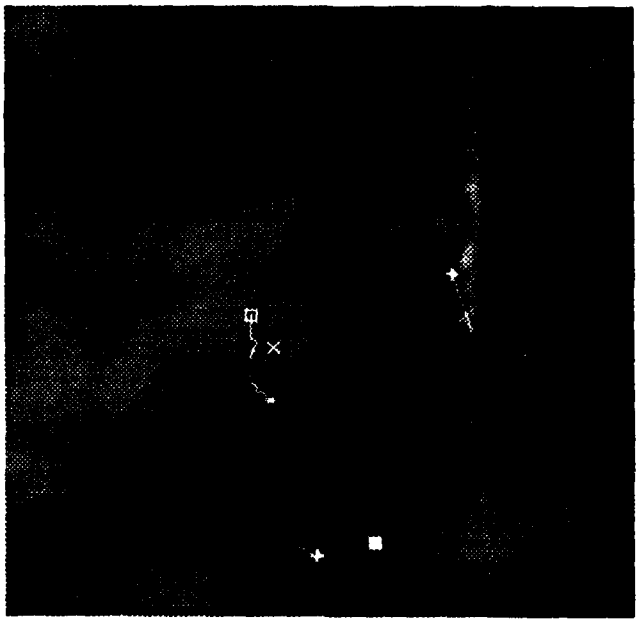


(7.a)

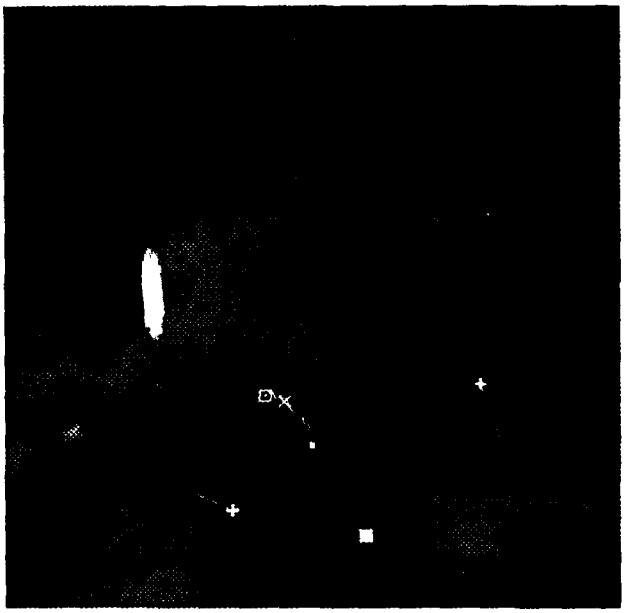


(7.c)

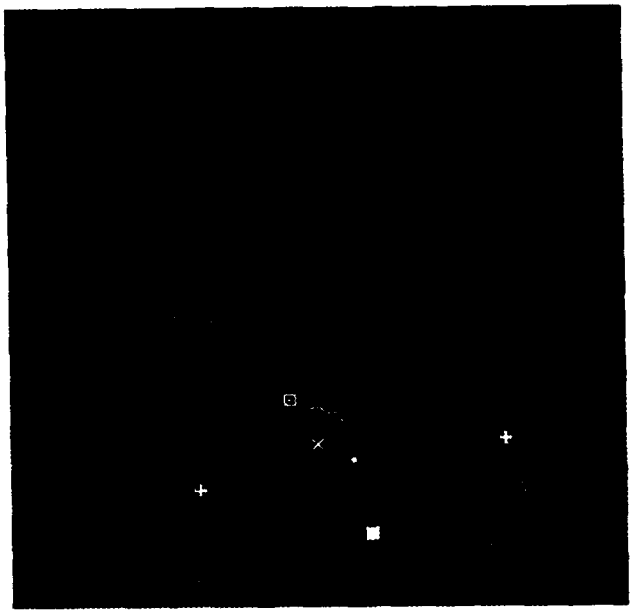
Figure 7: The path of each stage



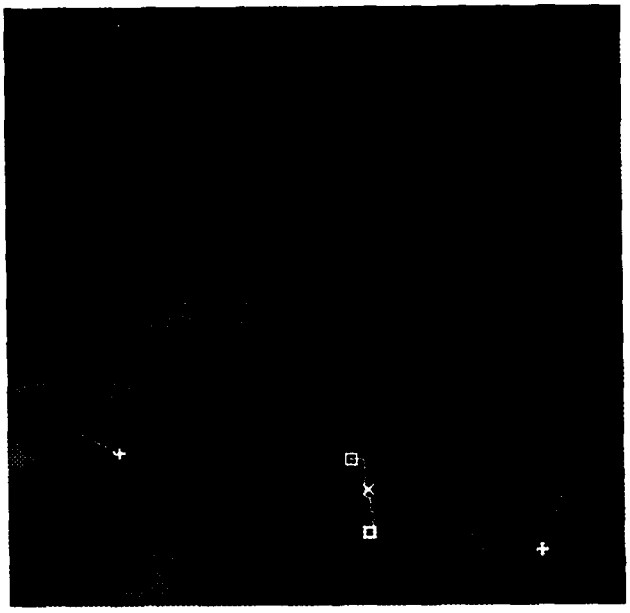
(7.b)



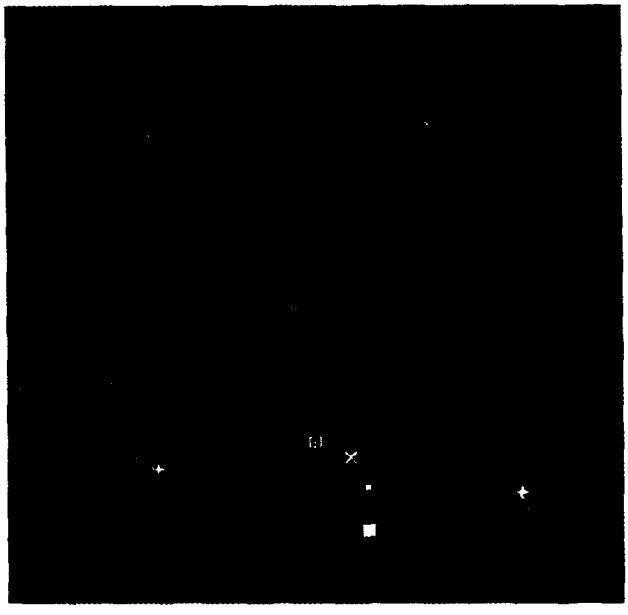
(7.d)



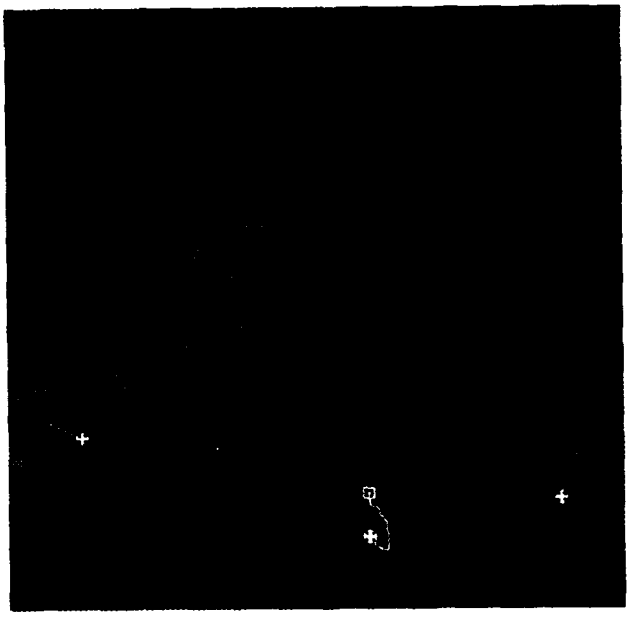
(7.e)



(7.g)



(7.f)



(7.h)

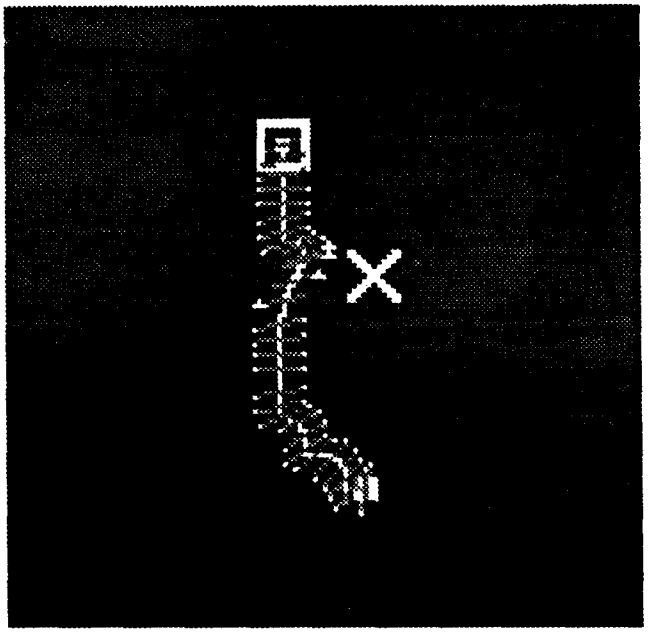


Figure 8: The configuration of the group and the actual location of each agent along the path of the second stage.

2. Obtaining the elevation data for the initial strip
   For each PE in the ray structure, its corresponding grid cell on the terrain can be computed in a way similar to that in the PRVA, and the elevation data can then be obtained by global communication. The grouping technique in our PRVA algorithm is applied to alleviate the congestion from concurrent reads. However, it cannot be eliminated completely: in the worst case, the number of concurrent reads is reduced to the range of u.
3. The strip-wise iterations
   The visibility analysis is conducted along all rays in parallel in the beginning of the loop. The results are combined into a count of visible destinations cells for each source cell on the strip. Special care should be taken in combining these results to avoid duplicated counting. This count is stored in the first active PE of the first active ray for each source cell. The ray structure is then updated for the analysis of the next strip. The steps are shown below:

```
Updating the ray structure for the next strip
begin
    de-activate the first active u-v slice;
    for all active PEs in the ray structure
        change the starting point of each ray to the
            next grid point along the sweeping direction;
        compute the new corresponding terrain
            cell;
        if it is different from the previous one
            if there is a u-neighbor previously
                corresponding to this terrain cell
            then
                get the elevation data from this
                    u-neighbor;
            else
                de-activate the entire ray the PE
                    is in;
        end if
    end for all
    de-activate rays whose emitting angles are
        greater than 45 degrees;
end
```

After the iterations, the counts of visible destination cells stored in the ray structure are sent back to the source cell on the terrain and are added to the results from other sweeps. It is an exclusive write operation since only one PE in the ray structure was designated to keep the count for each source cell and only these PEs will participate in this communication.

The resulting complexity is

$$O(L_S^2 L_D W \log W + L_S L_D W \times Comm_{read} + L_S^2 \times Comm_{write}) \text{ operations}$$

with a minimal

$$O(L_S \log W + Comm_{write} + Comm_{read}) \text{ time},$$

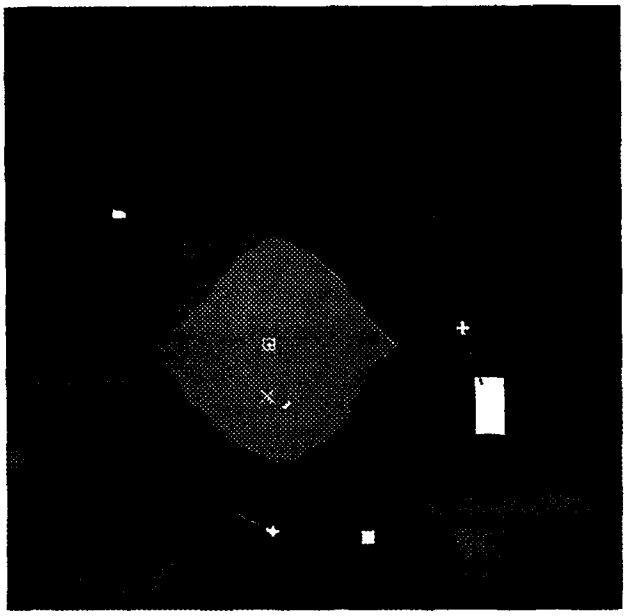
using up to $kL_S L_D W$ processors, where k is a constant determined by the number of far sides and the number of extended sources.

The bottleneck of the computation derives from global communication, and its reduction is achieved in three ways: the number of occurrences of global communication operations is reduced to only two (one for read and one for write), the total number of such operations is reduced by a factor of L_S, and the congestion due to concurrent read/write operations is minimized. The computation part is the same as the PRVA algorithm, since the same number of rays is allocated for each point and the computation along each ray remains the same. Experimental results showing the improvements due to the reduction in communication complexity can be found in [6].

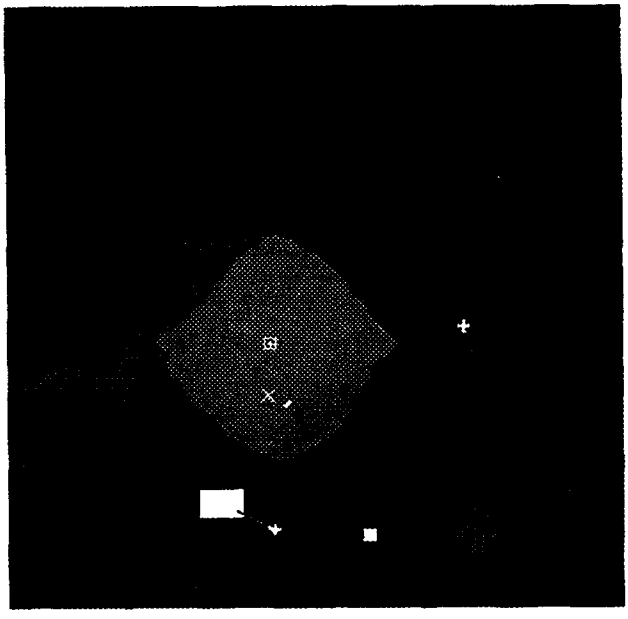
4 Implementation and Experimental Results

In this section, we present the results of our implementation of the algorithms. All experiments were conducted on a Connection Machine CM-2 using 8K processors. The terrain size is 512×512 grid cells, and the terrain cells are mapped to a two-dimensional array of virtual processors.

Figure 7 illustrates an example of the planning algorithm, which consists of eight subplans. Each subplan is illustrated by marks on the terrain map indicating the centers of the groups, and the grey level in the background shows the elevation at each pixel: brighter pixels are higher. The observer is indicated by an "X", and the initial location of the group being planned is marked by a square with a dot in its center. The line connecting the square to a small white dot is the path planned for the center of the group, and the small white dot at the end of the path indicates the subgoal location. The large solid white square near the bottom of each figure shows

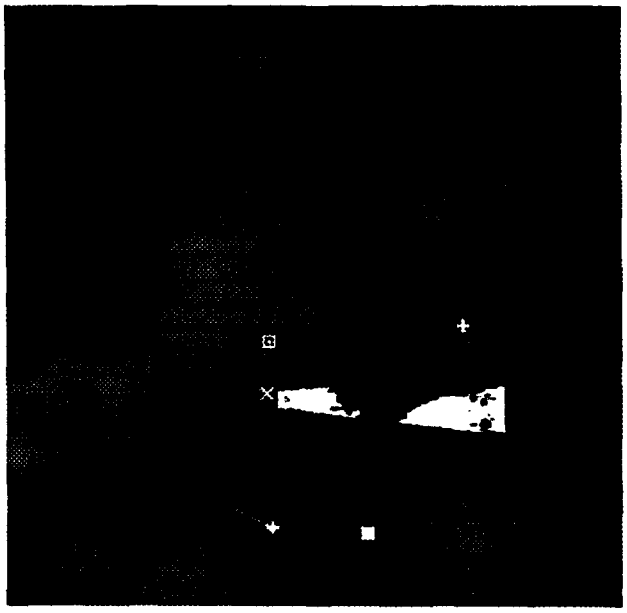


(a)



(b)

Figure 9: The source and destination regions in the region-to-region visibility analysis of the third stage.



(a)



(b)

Figure 10: The visibility from the vicinity of the subgoal to the destination regions in the third stage.

the location of the final goal. The adversary locations in the beginning of each subplan are indicated by crosses, and the emanating lines illustrate their predicted trajectories. In the figure for the first stage, both groups are located at the same initial position.

After finding the path for the center, the locations of the two agents are determined by the direction of motion and the spacing between agents. Figure 8 enlarges the region around the path of the second stage and shows the configuration of the group as line segments. The agents are located at the end points of each line segment. The safety and intervisibility of each agent are indicated by the brightness and the shape of the marks at the agent locations. A white dot means a path point at which the agent is safe and visible to its partner. A black dot means a path point at which the agent is neither safe nor visible to its partner. A white $\perp$ indicates an agent location that is safe but not visible to its partner, while a black $\perp$ indicates an agent location that is not safe but visible to its partner.

The key computation in the selection of the subgoal is the region-to-region visibility analysis. Figure 9 shows the regions considered in the region-to-region visibility analysis in the third stage. The source region is the small set of white points to the right of the observer (the "X" mark), which consists of the candidate cells for the subgoal. The destination region is the region spanned by the possible adversary trajectories of the next stage: a circumscribing upright rectangle is used. The subgoal is chosen using the visibility information. The visibility from the vicinity of the subgoal to the destination region is shown in Figure 10 by the white area. The experimental result shows that the planning system effectively returns a fairly good path.

References

[1] Guy E. Blelloch and James J. Little. Parallel solution to geometric problems on the scan model of computation. In *Proc. of the 1988 International Conference on Parallel Processing*, pages 218–222, 1988.

[2] John Canny. *The Complexity of Robot Motion Planning.* MIT Press, 1988.

[3] D. Hearn and M.P. Baker. *Computer Graphics.* Prentice-Hall, 1986.

[4] W. Daniel Hillis. *The Connection Machine.* MIT Press, 1985.

[5] John Reif and Micha Sharir. Motion planning in the presence of moving obstacles. In *Proc. 26th Symposium on Foundations of Computer Science*, pages 144–154, 1985.

[6] Y. Ansel Teng, Daniel DeMenthon, and Larry S. Davis. Region-to-region visibility analysis using massively parallel hypercube machines. Technical Report CAR-TR-578, Center for Automation Research, University of Maryland, 1991.

[7] Y. Ansel Teng, Daniel DeMenthon, and Larry S. Davis. Stealth terrain navigation. Technical Report CAR-TR-532, Center for Automation Research, University of Maryland, 1991.

Three Dimensional Object Tracking and Recognition Using a Dual-Drive Hybrid Controller*

Kelly A. Korzeniowski and William A. Wolovich

Laboratory for Engineering Man/Machine Systems
Division of Engineering,
Brown University,
Providence, RI 02912

Abstract

A robot that functions in real world situations must be equipped to handle randomness in its environment. The purpose of this research is to employ multiple control laws to enable an IBM 7565 robot to move through a constrained environment and recognize the objects it encounters. The robot is equipped with a force/torque sensor on its end effector. No prior information is used by the robot. While exploring the workspace, when the robot encounters an object, *dual – drive* force/velocity control and force feedback enable it to track each object's surface and collect position information that is used for recognition. Experimental results are presented.

1 Overview

Tactile sensing and force control have been employed to enable a robotic system to gather data about its environment in order to perform tasks and/or to identify objects. The research of Khatib [1] is concerned with real-time motion and force control of robot systems to accomplish finger sensing and assembly operations. The research of Allen [2] employs tactile sensing along with computer vision to recover the shape of an object by using sparse contact data points. Tactile sensing in [2] is directed by prior knowledge of the position of an object.

This paper will present a three dimensional object tracking algorithm that has been designed with the intent to be as object independent as possible. Experiments are conducted by placing different three dimensional objects in the workspace. The robot moves on a search path until an object is encountered. A force/velocity *dual – drive* controller is then implemented which moves the robot along the surface of the object without prior knowledge of its shape or configuration within the workspace. Planar slices of tracked points are collected in order to recognize an object.

The dual-drive control algorithm is implemented on an IBM 7565 robot equipped with a six axis strain gauge sensor on its end effector. Force sensor data has an advantage over machine vision in certain environments; i.e. when the lighting is poor or if the work area is cluttered. These situations make it difficult to extract reliable information from vision algorithms.

The tracking algorithm is tested using simple and complex real-world objects. The tracking method facilitates the collection of dense data sets that can be used for object recognition.

*This work was partially funded by the NSF in conjunction with the Advanced Research Projects Agency of the Department of Defence under Contract No. IRI-8905436

2 Prior Results

SIERA (System for Implementing and Evaluating Robotic Algorithms) provides the research environment for the IBM 7565 [3]. The system allows the user to interactively modify any robot related function and change the control law to suit the current task. In *SIERA*, a separate computer system is used for each task to best match the hardware to the task requirement. Real-time processing is done on the bus-based *RTSS* (Real Time Servo System). All controller modifications run on the *RTSS*. The fast processing time enables the robot to respond quickly to force feedback information. When more computational power is required, such as for graphic displays and/or object classification programs, the link-based Armstrong Multiprocessor is used [4].

Multiple control laws are employed to enable the robot to perform functions in its workspace. The controllers are supervised by an *observer* program. The *observer* is in charge of activating real-time controllers according to force feedback information.

When the robot is tracking an object, a *dual-drive force/velocity controller* is active [5]. Force feedback is used to determine the surface normal and velocity feedback for the surface tangent. The dual-drive controller defines the signs of the vectors relative to a point inside the tracking path. In Figure 1, motion is defined to be positive when the vector $\mathbf{V}$ is counterclockwise with respect to the inner point P. The force vector $\mathbf{F}$ is positive when it is in the same direction as $\mathbf{R}$. The relation between the velocity and the force is orthogonal if the tracking surface is sufficiently stiff and the friction between the surface of the object and the end effector is negligible. If the surface is frictionless, the normal will be in the same direction as the force vector $\mathbf{F}$. Using force sensor readings, the outward unit normal and unit tangent vectors are given by,

$$\mathbf{N} = \frac{1}{|\mathbf{F}|}(\mathbf{F}_x, \mathbf{F}_y)$$

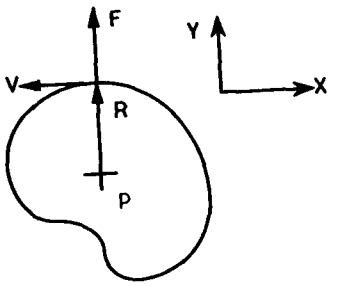


Figure 1: Surface Tracking with the Dual-Drive Controller

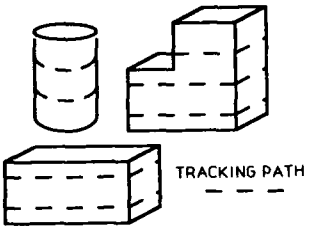


Figure 2: Planar Tracking Paths

and

$$\mathbf{T} = \frac{1}{|\mathbf{F}|}(-\mathbf{F}_y, \mathbf{F}_x).$$

In the case of a stiff surface, the velocity due to force corrections is small and $\mathbf{V}$ is in the tangential direction. The outward unit normal and unit tangent vectors are given by,

$$\mathbf{N} = \frac{1}{|\mathbf{V}|}(\mathbf{V}_y, -\mathbf{V}_x)$$

and

$$\mathbf{T} = \frac{1}{|\mathbf{V}|}(\mathbf{V}_x, \mathbf{V}_y),$$

where $|\mathbf{F}| = \sqrt{\mathbf{F}_x^2 + \mathbf{F}_y^2}$ and $|\mathbf{V}| = \sqrt{\mathbf{V}_x^2 + \mathbf{V}_y^2}$. These four equations for the unit normal and unit tangent vectors give four different dual-drive implementations [5]. When the inner point P cannot be specified, the sign of the velocity and force vectors are computed by noting the force at the end effector and the actual trajectory of the robot. This is necessary in applications where the shape of the object, and thus the complete tracking path, are unknown. [6].

When the robot is in the dual-drive mode, the controller is complemented by a *force − check* controller [6]. The force threshold check sets an interrupt when the robot looses contact while tracking. This is likely to happen when the robot encounters a sudden change in surface orientation, i.e. and edge. When the interrupt is set, the dual-drive controller is switched to a surface searching routine. The robot moves in an outward turning square spiral until contact is regained. The angle between the surface normal and the end effector is checked and if necessary, the end effector is reoriented, at which time surface following continues. The stored position feedback signals that are collected during the tracking can be used by a classifier program to identify the shape and location of an object.

The controller *Fcomply* is active when the robot is moving along a specified "search" trajectory, and it is not in contact with a surface [5]. *Fcomply* is a force compliant damping controller. When an object is encountered, the damping controller converts a desired force value, F_d, to a velocity vector, $\mathbf{V}$, by a damping constant, B.

$$\mathbf{V} = \frac{F_d}{B}$$

The *Fcomply* controller also has velocity and position saturation to keep the robot at a safe velocity level within the workspace. When *Fcomply* is active, the robot moves along some predescribed path along a trajectory until a force threshold is exceeded, which results when the end effector comes in contact with an object.

These controllers have been successfully employed for a two dimensional object recognition demonstration [6]. In such experiments, the IBM 7565 moves on a prespecified rectangular path through the workspace which enables the robot to sufficiently explore the entire environment. Objects such as a triangle, a quadrilateral and an ellipse are placed in the workspace. No prior information about the shape or the location is given. Dual-drive control enables the robot to track and to recognize the objects it encounters.

3 Current Research

The current focus of this research has been directed at expanding the two dimensional algorithm to enable the robot to track and to recognize three dimensional objects. Initially, the objects considered are spheres, cylinders, cubes, cones and variations of these. These objects are chosen because complex, real world objects are often composed of these basic shapes.

The three dimensional object tracking algorithm is meant to be as general as possible. It does not depend on an object's orientation, shape or location in the workspace; i.e. there is no prior knowledge of the object's shape or surface contour. In our initial experiments, all objects are tracked using the two dimensional tracking algorithm. The robot moves along an object's surface in a plane parallel to the earth's surface, and planar slices are taken at increasing levels in the vertical $+Z$ direction. Some examples of these simple objects are shown in Figure 2. The location of the object and the change in shape of each planar slice is identified from the information gathered during the side tracking. Information about the change in the surface contour in the vertical

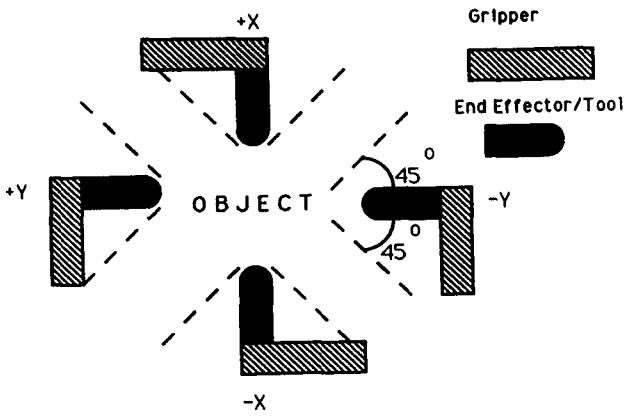


Figure 3: Configurations of Gripper/End Effector

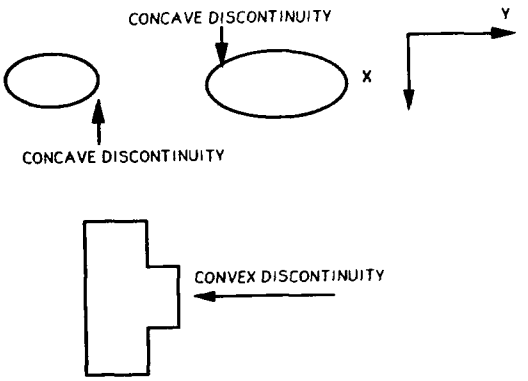


Figure 4: Planar Slices with Concave and Convex Discontinuities (Top View)

$+Z$, or any direction, can be extrapolated from explicit knowledge of the X, Y and Z position for each planar slice. Given this information, the recognition program is able to distinguish between simple objects, such as spheres, cylinders, cones, and boxes.

Ideally the robot should move along the surface of an object with the end effector oriented along the surface normal, since this orientation implies minimal interference from friction. However, it is difficult to implement such a controller. The present dual-drive controller must be reconfigured each time the orientation of the end effector changes, and in the case of a cylinder the end effector would have to be constantly reoriented. A safe threshold for the magnitude of the angle between the tool and the surface normal, $\mid \theta \mid$, was found to be $\mid \theta \mid < 45°$. The angle $\mid \theta \mid$ is inferred from the force sensor readings and the trajectory of the robot. Figure 3 illustrates how the robot can track around an object using four basic configurations. Therefore, an object is tracked with the end effector in one configuration until $\mid \theta \mid \geq 45°$, at which point the end effector is reconfigured for tracking to continue. In the workspace, these configurations point the end effector in either the $+Y$, $-X$, $-Y$ or $+X$ direction. Each of these basic configurations requires a different version of the dual-drive controller.

For three dimensional object tracking, when the robot encounters an object in the workspace, it follows the *side* in the dual-drive mode with the end effector in one of the basic configurations. At an *edge*, the magnitude of θ is such that the end effector must be reoriented for tracking to continue. At this time the dual-drive controller is deactivated, and the robot moves away from the side and reconfigures the end effector. The *Fcomply* controller moves the robot toward the new side until contact is made, then an appropriate dual-drive controller is reactivated. This continues until the robot has completely encircled the object in the X-Y plane, i.e. when the robot has tracked around the object in all four configurations. The robot then moves a prespecified amount in the $+Z$ direction and repeats the process.

It should be noted that in order to insure that the gripper does not collide with an object, there is a limit to the amount the robot arm may travel when approaching a side. The measure is equal to the length of the end effector (see Figure 3). Position information is available while the robot is in contact with an object, and the point where the end effector begins the approach to a side (x, y) is recorded for each configuration. If contact is not make during an approach, the robot attempts to make contact in the next configuration. The approach begins at the (x, y) position for that configuration, which was recorded during the previous tracking pass. When the robot is unable to make contact with the object, because the location of the end effector exceeds the height of the object in all of the four basic configurations, tracking in the X-Y plane is completed.

Once the robot is able to track simple three dimensional objects, more realistic, complex objects will be employed. There are certain problems associated with the identification of many of these real objects. In particular, each of the four basic configurations of the end effector represents a progressive 90° increase from the previous configuration. For example, when moving counterclockwise the object's surface may be followed from $+Y$ to $-X$ ..., $+X$ to $+Y$ ..., etc. depending on the starting point. Obviously not all objects have surface normals whose angle *only* increases in this way. The algorithm must deal with *concave* and *convex* discontinuities on the object's surface. A concave discontinuity is defined as a temporary increase in θ for a *side*. It can be thought of as an indentation in the planar slice (see Figure 4). A convex discontinuity is defined as a temporary decrease in θ for a *side*, $\theta < -45°$ (see Figure 4). Due to size considerations, the end effector cannot be reoriented to explore a concave or convex discontinuity on the surface normal. Therefore, when the sensors indicate that the end effector has come to one of these surface discontinuities, the active dual-drive controller is deactivated and the robot switches to an appropriate alternate tracking scheme. In the case of a concave discontinuity, the robot moves on a trajectory across the

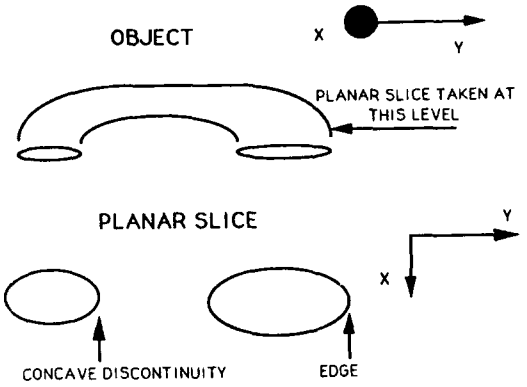


Figure 5: Concave Discontinuities *V.S.* Edges

area until contact is made with the other side. It then switches to a routine that searches for the point where tracking can continue in the dual-drive mode. Using *Fcomply*, the robot moves in an outward turning spiral until contact is made. This contact point is tested to find the angle the surface makes with the end effector. If the angle is acceptable, the dual-drive controller is reactivated and surface tracking continues. If not, the action is repeated until either the search is successful or the robot has reached an *edge*. The robot handles a convex discontinuity in an analogous fashion. At the beginning and at the end of the convex area, the robot searches for a point where it can safely resume tracking with the dual-drive controller.

A convex discontinuity, and the point where the end effector must be reconfigured; i.e. an *edge*, are both indicated when $\theta \geq 45°$. Figure 5 illustrates this point. Obviously, successful tracking depends on the ability of the robot to distinguish between these two instances. In order to insure that a concave discontinuity is always properly distinguished from an *edge*, the outer limits of the object are assumed to be known before the object is actually tracked, the justification being that in the future this information will be obtained from a machine vision algorithm. Stereo machine vision cannot provide the dense surface descriptions that tracking yields. However, the sparse matches provided by stereo algorithms are reliable [7]. These sparse data matches give an outline of the object. In this way, stereo machine vision complements the tracking algorithm by giving a first estimate of the area that the robot must track in each basic configuration.

4 Implementing the Algorithm

Simple three dimensional objects have been used to test the tracking algorithm. An illustration of the objects and the actual data points collected are shown in Figure 6. The arrows on the illustration indicate the levels at which the planar slices were taken.

In order to show that the tracking scheme gathers ample data points to identify a simple object, the data sets

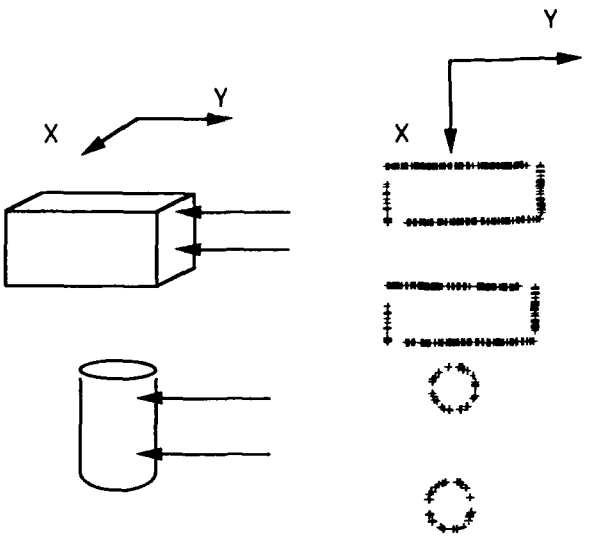


Figure 6: Simple Objects and Actual Data Points Collected

shown in Figure 6 were used as input to a shape classifier program. The planar slices from the box were identified as quadrilaterals and the data points from the cylinder were recognized as circles. Hence, simple objects like those shown in Figure 2 can be recognized by combining shape information about the planar data slices.

Complex objects have also been employed in the experiments. A standard telephone receiver was placed at different orientations in the workspace. Figure 7 and Figure 8 show the actual data collected during surface tracking. Again note the arrows on the illustration. In the future the data collected will be used to recognize the object.

5 Conclusions and Future Plans

This paper outlines a robotic object tracking algorithm which employs multiple controllers. The *dual − drive* force/velocity controller is used when the robot is in contact with an object, and the damping controller, *Fcomply*, is employed to move the robot on a specified trajectory. The combination of these controllers enables the robot to explore its environment and gather data about objects in the workspace.

The algorithm has been successfully implemented on an IBM 7565 robot, equipped with strain gauges on its end effector. Dense data sets have been collected during surface tracking with the intention that this information will be used to recognize various objects. These data sets have been used to recognize simple objects such as a box and a cylinder.

With the additional feature of recognizing *concave* and *convex* discontinuities on an object's surface, the algorithm can be applied to complex objects. The perfor-

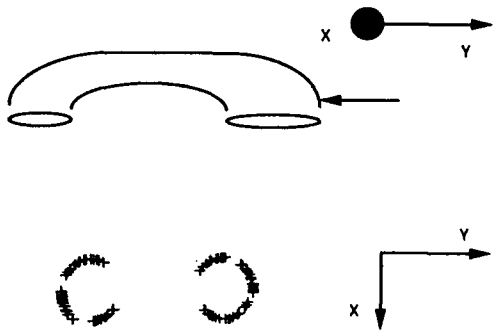


Figure 7: Tracking the Telephone Receiver: Actual Data Points

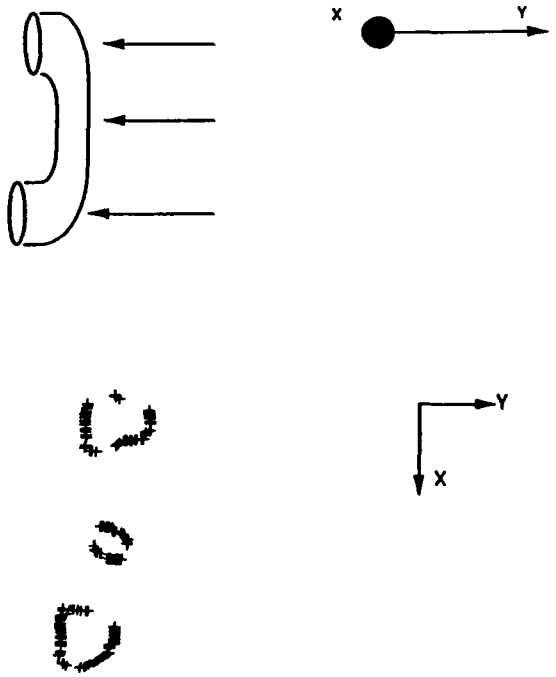


Figure 8: Tracking the Telephone Receiver: Actual Data Points

mance of the algorithm is not dependent on the orientation of an object. A telephone receiver was placed at different orientations in the workspace, and tracking was successfully demonstrated in all cases.

Future investigations in the area of *dual–drive* control will concentrate on developing a force/velocity controller that will move the robot along a trajectory in three dimensions and thus enhance the data collection ability of the robot. The necessary reconfigurations of the end effector should be made while the tool remains in contact with the surface. This will eliminate the discontinuous motion presently associated with reorienting the end effector and result in faster tracking. These enhancements will enable the robot to follow a complex path along the surface of an object and limit the uncertainty involved in making an identification.

Work is underway to enable the recognition program to identify complex shapes, much like the planar slices taken from the telephone receiver. Here there is the possibility of including information about the "features" recognized by the algorithm, i.e. edges, concave and convex discontinuities, in order to give an indication of the identity and orientation of an object. Work is progressing toward incorporating machine vision in an interactive object recognition system that will request additional tracking data from a specified area of the object in order to increase the probability of making a positive identification. When these enhancements are employed as part of the complete tracking system, the robot will be able to navigate in a more complex, real-world environment.

References

[1] Khatib, O., "A Unified Approach for Motion and Force Control of Robotic Manipulators: The Operational Space Formulation", *IEEE Journal of Robotics and Automation*, vol. 3, no. 1, 1987, pp. 43-53.

[2] Allen, P.K., Roberts, K.S., "Haptic Object Recognition Using a Multi-Fingered Dextrous Hand", *IEEE Conference on Robotics and Automation*, Scottsdale, AZ, May 1989.

[3] Kazanzides, P., Wasti, H., Weinstein, R., and Wolovich, W.A., "SIERA: System for Implementing and Evaluating Robotic Algorithms", *Brown University Technical Report LEMS-23*, June 1986.

[4] Rayfield, J.T., "Armstrong, a Loosely-Coupled Multiprocessor Testbed for Reconfigurable Topologies", *Ph.D. Thesis, Brown University*, May 1988.

[5] Kazanzides, P., "Multiprocessor Control of Robotic Manipulators", *Ph.D. Thesis, Brown University Technical Report LEMS-47*, May 1988.

[6] Bradley, N.S., "Force Control Methods for Environment Interaction and Recognition", *Master's Thesis, Brown University*, May 1990.

[7] Allen, P.K., *"Robotic Object Recognition Using Vision and Touch"*, Kluwer Academic Publishers, Norwell, MA, 1987.

Towards an Assembly Plan from Observation: Task recognition with polyhedral objects*

Katsushi Ikeuchi and Takashi Suehiro
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Currently, most robot programming is done either by manual programming or by the "teach-by-showing" method using a teach pendant. Both of these methods have been found to have several drawbacks.

We propose a novel method to program a robot, the assembly-plan-from-observation (APO) method. The APO method aims to build a system that has the capability of observing a human performing an assembly task, understanding the task based on the observation, and generating the robot program to achieve the same task.

In particular, this paper defines assembly relations which serve as the basic representation of each assembly task. Then, we verify that such assembly relations can be recovered from the observation of human assembly tasks, and that from such assembly relations, it is possible to generate robot motion commands to repeat the same assembly task. Finally, we demonstrate an APO system based on the assembly relations.

1 Introduction

The key characteristic of robots is their versatility. They can be used to perform a large variety of tasks without a major re-design of the robot. This versatility is due to the generality of the robot's physical structure, but a robot's generality can be exploited only if the robot can be easily programmed.

Several methods to program a robot have been proposed. Such methods include: teach-by-showing, teleoperation [17, 12, 3], textual programming[2], and automatic programming [6, 9, 7]. In teach-by-showing methods, an engineer stores, using a teach pendant in teaching mode, a path along which a robot should move repeatedly. In run mode, the robot follows the path it was previously taught. This is the most common method to program a robot in industrial applications. This method is suitable for programming a robot to repeat simple movements. Moreover, this method is excellent because a robot can learn complicated paths from a trained engineer. However, this method requires that an engineer is in the same environment as the robot. Thus, we cannot use this method in hazardous environments such as in nuclear plants, underwater, or in outer space.

To remedy this problem, teleoperation methods have been proposed. This method uses a master manipulator for teaching and a slave manipulator for execution. An engineer controls the master manipulator in a safe environment while monitoring the hazardous environment with a remote TV camera and display. The slave manipulator in the hazardous environment executes real operations based on control signals from its master manipulator. Since this method does not require an operator in the execution environment, it is suitable for the operation in hazardous environments. However, by using this method, we can only teach a robot trajectory information. It is difficult to build a flexible robot system able to use force control with error recovery capabilities. It is also true that we have to reconstruct entire programs, even when a very minor change in the program is desired.

Textual programming is often used in academic environments. A programmer stores a robot command sequence in a computer as a textual program. By using a compiler or an interpreter, a command sequence in a textual program is converted into a form that the robot can execute. This method is quite flexible because we can store any kind of control programs. However, it requires a long development period and expert programmers.

In order to speed up the programming process, automatic programming has been proposed. The method tries to develop geometric reasoning systems which can generate textual programs to control a robot from geometric information given by geometric models and task specifications. This direction is quite promising, however, there are many issues to be addressed before we have a complete automatic programming system. Such issues include: how to generate a sequence of operations, how to determine a grasp point for each operation, how to determine a global path to move an object while avoiding collisions with other objects. It is quite difficult to build a complete automatic programming system, though perhaps not impossible.

We propose a novel method that combines automatic programming and teleoperation. We propose to add a vision capability that will observe human operations to an automatic programming system (a geometric reasoner). In particular, we propose a system that observes a human performing an assembly tasks while a geometric reasoner analyzes and recognizes such tasks from observation, and generates the same assembly sequence for a robot. We will refer to this paradigm as Assembly Plan from Observation (APO).

Due to the geometric reasoning capability, the APO system understands the operations that the operator is performing. Thus, the system for example can discard unnecessary motions which are often introduced by a human teleoperator. The

*This research was sponsored by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C-1465 Order No. 7597.

system can also insert error recovery routines into the generated assembly plans. In this regard, APO is superior to the teleoperation method.

Due to the vision capability, the system can solve several otherwise extremely difficult problems, such as path planning and determining the optimal assembly sequence, by simply observing a human performing the operation. In this regard, APO is superior to the automatic programming method.

2 Assembly plan from observation

In an APO system, a human operator performs assembly tasks in front of a video camera. From the camera, the system obtains a continuous sequence of images recording the assembly tasks. In order for the system to recognize assembly tasks from the sequence of images, the system has to perform the following six operations (See Figure 1.):

- *Temporal Segmentation* - dividing the continuous sequence of images into meaningful segments which correspond to separate human assembly tasks,
- *Object Recognition* - recognizing objects and determining object configurations in a given image segment.
- *Task Recognition* - recognizing assembly tasks based on the results of an object recognition system.
- *Grasp Recognition* - recognizing where and how the human operator grasps an object for achieving the assembly task.
- *Global Path Recognition* - recognizing the path along which the human operator moves an object while avoiding collision.
- *Task Instantiation* - collecting necessary parameters from object recognition, grasp recognition, and global path recognition results for performing the recognized assembly tasks, and setting up assembly plans to perform the same task using a robot manipulator.

In this paper, we will concentrate on the task recognition and task instantiation modules, because these two parts form the main loop for the assembly plan from observation.

The outline of the modules are as follows:

Our object recognition module identifies each object using the object models from a given image segment. The module represents the recognition results in a world model, as shown in Figure 1, by using the geometric modeler, Vantage.

Our task recognition module recognizes object relations in two image segments and extracts the transition between two object relations from the two segments. The task recognition system has *abstract task models* in a data base. Each abstract task in the data base describes a transition between two different object relations. From the task models in the data base, the system identifies a task model that describe the transition needed to achieve the observed object relations, as shown in Figure 1.

Our task instantiation module represents the recognition result as an instantiated task model. An instantiated task model associates a transition with an action capable of causing the transition. It also includes appropriate parameters to achieve the action based on the given scenes. Such parameters include object locations and the grasping locations for the action. The instantiated task model also includes the global path along

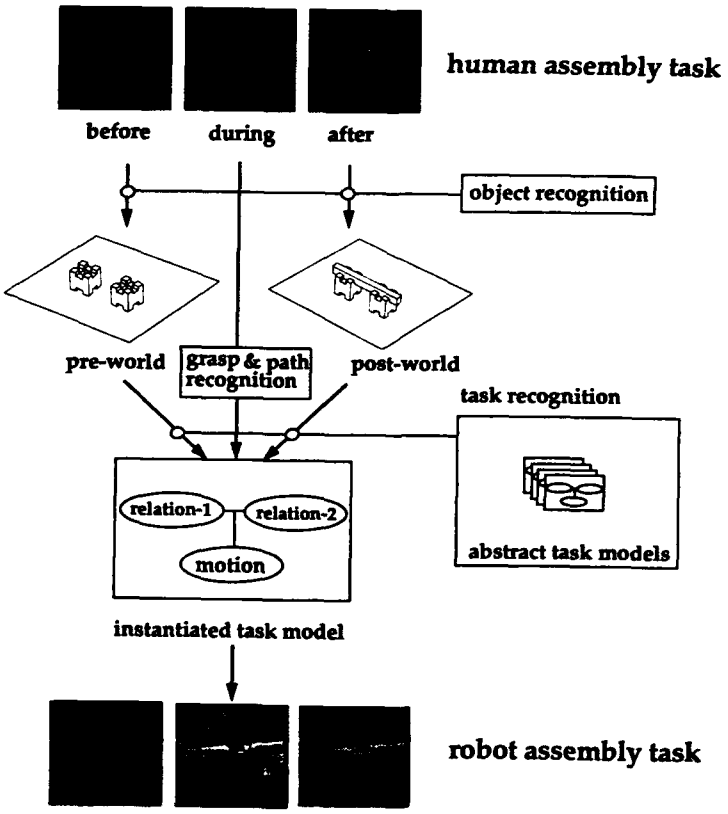


Figure 1: Assembly plan from observation.

which to move an object. The system, then, inserts the obtained grasp and stack locations into the command sequence. Finally, the command sequence is sent to the robot.

3 Defining Task Models

In order to develop task models for an APO system, we have to define representations to describe assembly tasks. In this section, we will define *assembly relations* for such representations. Then, we will examine that such assembly relations satisfy the two requirements.

- *recoverability* - assembly relations can be extracted from observation,
- *inferability* - a human assembly task can be inferred from an assembly relation, and it is possible to generate assembly operations for a manipulator from the assembly relation.

Finally, we will consider how to define assembly task models using the assembly relations.

3.1 Assembly relation

In each assembly task, at least one object is manipulated. We will refer to the object as the *manipulated* object. The manipulated object is attached to other stationary objects, which we refer to as *environmental* objects, so that the manipulated object achieves a particular relation with environmental objects.

We will define assembly relations with respect to face contacts between a manipulated object and its stationary environmental objects. The essential goal of an assembly task is

to establish a new face contact between a manipulated object and environmental objects. For example, the goal of a peg-insertion is to achieve face contacts at the side and bottom faces of the peg against the side and bottom faces of the hole. Thus, it is effective to use face contact relations as the central representation for defining assembly task models.

To make the overall problem manageable, we concentrate on a world of polyhedral objects in which only one polyhedron may be moved by one assembly task. An assembly relation will be defined between a manipulated polyhedron and several stationary environmental polyhedra. This restriction still leaves a diverse range of interesting relationships, actions, and resulting assemblies.

Such face contact relations satisfy the recoverability requirement.

- *Face contact relations can be obtained by analyzing geometric models.* An object recognition program, such as in [4] can recognize a manipulated object, determine its configuration, and represent the recognition result in a geometric modeler, as well as the geometric representations of other stationary environmental objects. By examining each face pair between the manipulated and environmental objects, new face contact relations can be determined as they occur.

Face contact relations also satisfy the inferability requirement.

- *Each face contact relation constrains possible motions.* At contacting faces, the orientations of surface normals are suffice for characterizing relative object movement constraints. For example, consider a box resting on a table. At the contact faces surface normals are parallel and opposing. In this position the box can only move up or parallel to the table. A more constraining case is a square bar inserted in a matching shaped hole. The bar's four faces contact their hole counterparts with opposing normals and the only possible motion lies along the hole's axis. Thus, from a face contact relation, it is possible to infer the assembly actions that cause such face contact relations.
- *Face contact relations characterize a control strategy necessary to maintain such relations.* Each face contact relation provides a constraint to motion. As long as the motion constraint is constant, the same mode of control is applicable. When the motion constraint changes, a different mode of control is required. For example, let us consider a box to be placed on a table and then slide on the table. Position control can be used to lower the box towards the table while the box is in the air (the box does not have any face contact.). When the box is about to make contact with the table (about to have one-face contact), force control is necessary to detect the collision which ensures that the box is on the table. Combined force and position control is necessary to slide the box on the table (for maintaining one-face contact). Face contact relations have been found to characterize required control strategies [14, 13]. Thus, such face contact relations can be used to determine a control strategy necessary to achieve such face contact relations in assembly actions.

Using such face contact relations as the basic representations, we will describe an assembly task with a transition between pre-assembly relations and post-assembly relations. Based on the description, we will build an APO system in the following steps:

- classifying all possible face contact relations (assembly relations) between manipulated and environmental objects,
- considering what kinds of transitions in assembly relations occur and building a tree in which each branch corresponds to one possible transition and each leaf node corresponds to an assembly relation, and
- assigning manipulator motions to achieve such assembly relation transitions (the completed tree is referred to as a procedure tree).

3.2 Taxonomy for Assembly Relation

For geometric objects in a polyhedral world, our taxonomy identifies all possible assembly relations based on the directions of contact surface normals. First, we will analyze a two-dimensional polygonal world and then a three-dimensional polyhedral world. Some related issues are found in [10].

3.2.1 Two-dimensional cases.

Assembly relations will be considered between polygons (not polyhedra) by using normal directions at contact edges. Figure 3.2.1 shows an assembly relation having unidirectional contact. Even if polygons have several contact edges with the same normal direction, they are considered as having unidirectional contact.

Normal direction of an edge can be represented as a point on the Gaussian circle by translating the unit normal so that its starting point sits at the origin of the coordinate system. Its tail then lies on the unit circle whose center is the origin. This mapping is referred to as a Gauss mapping. Equivalently, possible movement directions of the object polygon can be represented on the Gaussian circle. Points on the semicircle around the contact direction correspond to the possible movement directions of the object polygon. Points on the other semicircle correspond to the prohibited motion directions.

Bidirectional contact has two possible assembly relations as shown in Figure 3. Assembly relation 2d-b in Figure 3 has two maps located opposite one another on the circle. This case has two possible movement directions on the Gaussian circle. Relation 2d-c in Figure 3 has two oblique contact directions on the Gaussian circle. This case has several possible movement directions corresponding to a small arc.

Tridirectional contact at first seems to have three cases: relation 2d-d, relation 2d-e, and relation 2d-f, as shown in Figure 4. Relation 2d-d has two opposite points and one intermediate point; resulting in only one possible movement direction. Relation 2d-e has three points whose maximum arc is larger than π, allowing no movement of the object at all.

Relation 2d-f has three arbitrary points whose maximum arc is less that π. As Figure 4(c) illustrates, the middle contact direction does not affect the possible movement directions on the Gaussian circle. Thus, relation 2d-f is considered equivalent to relation 2d-c and is not considered an independent relation. When a relation has more than three directions of contact, it can be mapped to one of the relations mentioned above.

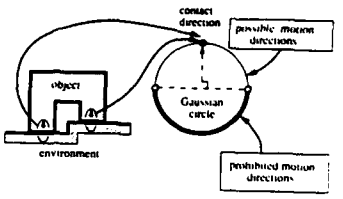


Figure 2: Unidirectional contact.

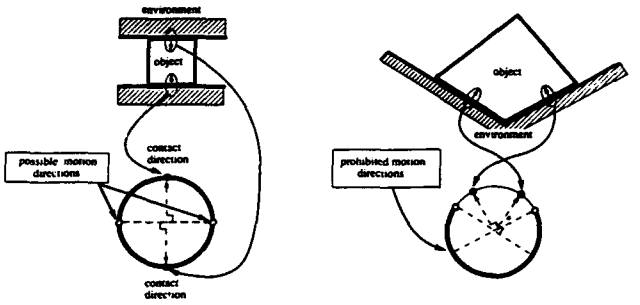


Figure 3: Bidirectional contact: (a) relation 2d-b; (b) relation 2d-c.

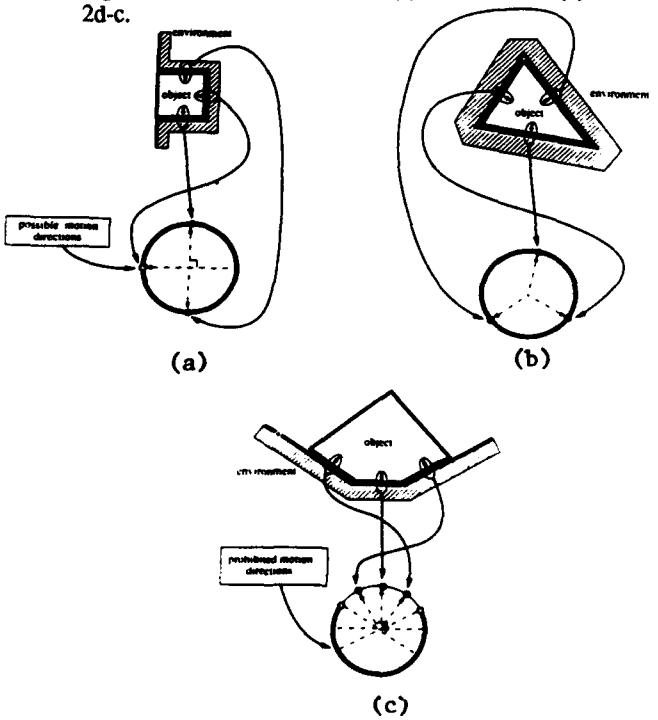


Figure 4: Tridirectional contact: (a) relation 2d-d; (b) relation 2d-e; (c) relation 2d-f. This is equivalent to relation 2d-c in terms of possible movement directions, and thus, is not considered an independent relation.

Table 1: 3D assembly relations

Class	Relation	Explanation
uni	3d-a	3d-a in Figure 5 shows unidirectional contact. The contact direction is represented as a point on the Gaussian sphere. Let us suppose the contact direction is mapped to the north pole of the Gaussian sphere. The possible directions of object motion can be represented as the northern hemisphere of the Gaussian sphere; the prohibited directions can be represented as the southern hemisphere.
bi	3d-b 3d-c	Bidirectional contacts have two different relations: 3d-b and 3d-c, depending on whether contact directions are in opposite directions or not. See Figure 5. Relation 3d-b has possible movements represented as a great circle on the Gaussian sphere. Relation 3d-c, on the other hand, has possible movements represented as an area bounded by two great circles on the Gaussian sphere.
tri	3d-d 3d-e 3d-f	The tridirectional contacts have three different relations: 3d-d, 3d-e, and 3d-f. Relation 3d-d and relation 3d-e have three coplanar contact directions. Relation 3d-d has possible movements corresponding to a great semi circle. Relation 3d-e has possible movements corresponding to two points. Between relation 3d-e and relation 3d-d, there exists a relation which has identical possible movement directions as relation 3d-c. Relation 3d-f has possible movements corresponding to a spherical area bounded by three great circles.
tetra	3d-g 3d-h	Relation 3d-g has possible movements corresponding to an arc, while relation 3d-h has one possible movement. Adding one more contact direction to relation 3d-e gives relation 3d-h.
hexa	3d-i	3d-i in Figure 5 has no possible movement directions.

3.2.2 Three-dimensional cases.

The same analysis can be applied to 3-D cases. 3-D cases consider the relationship among polyhedra. Table 1 summarizes the analysis of face contacts among polyhedra. The taxonomy has classes of uni-, bi-, tri-, tetra-, and hexadirectional contacts. Nine different contact patterns are extracted from this analysis.

We will represent the contact directions and possible movement directions on the Gaussian sphere as shown in Figure 5. The shaded areas indicate the prohibited movement directions of the object with respect to the environment. The non-shaded areas indicate the possible movement directions.

3.3 Assembly relation transitions

We will consider a sequence of manipulator operations to achieve each assembly relation from assembly relation 3d-s. Such a sequence of manipulator operations is grouped into a motion macro, i.e., a template of manipulator operations, which, when applied to an object, yields the desired assembly relation. This is possible because each assembly relation is defined so that we can apply the same manipulator control strategy to achieve the relation by changing only controller parameters, not the strategy.

In order to reduce the number of necessary templates, we will analyze each assembly relation in an iterative manner. We will analyze simpler relations earlier and more complicated relations later. Also, instead of considering a template to directly achieve a complicated relation from 3d-s, we will

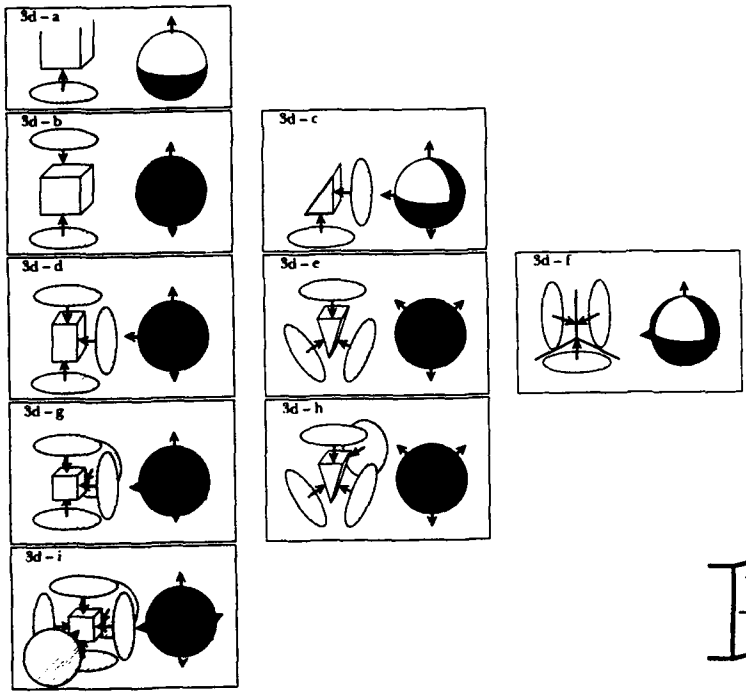


Figure 5: 3-D assembly relation taxonomy.

consider an intermediate relation, and then try to achieve the complicated relation. First, we try to achieve an intermediate relation from 3d-s by using the templates already considered. Then we try to achieve the final relation from the intermediate relation using a newly considered template.

In order to find an appropriate intermediate relation, for each assembly relation, we consider disassembly actions from the assembly relation, and extract all possible immediate intermediate assembly relations just prior to the assembly relation. We do this because considering disassembly actions is easier than considering assembly actions.

Several intermediate relations sometimes occur from the same assembly relation due to 1) the variation in shapes of contact faces, and 2) the variety of possible disassembly operations.

In case that due to variations in the shapes of contact faces, we have to analyze all intermediate relations and assign appropriate motion templates to all transitions from the intermediate relations to the desired relation.

In case that due to the variety of possible disassembly operations, we can choose one appropriate intermediate relation among the several intermediate relations. We choose the one which is achieved by the simplest and most robust operation under uncertainty in positional information. In order to select such intermediate relation, we use the following criteria:

1. In the case that a *direct detach* motion (a motion which immediately breaks a face-contact) exists, choose it.
2. In the case that a *lateral* motion (a motion maintain the same contact relation) that would break face-contacts by crossing a certain boundary exist, choose it.
3. In the case in which several candidate motions satisfy criterion 1 or criterion 2, choose the motion which least reduces the number of face contacts.

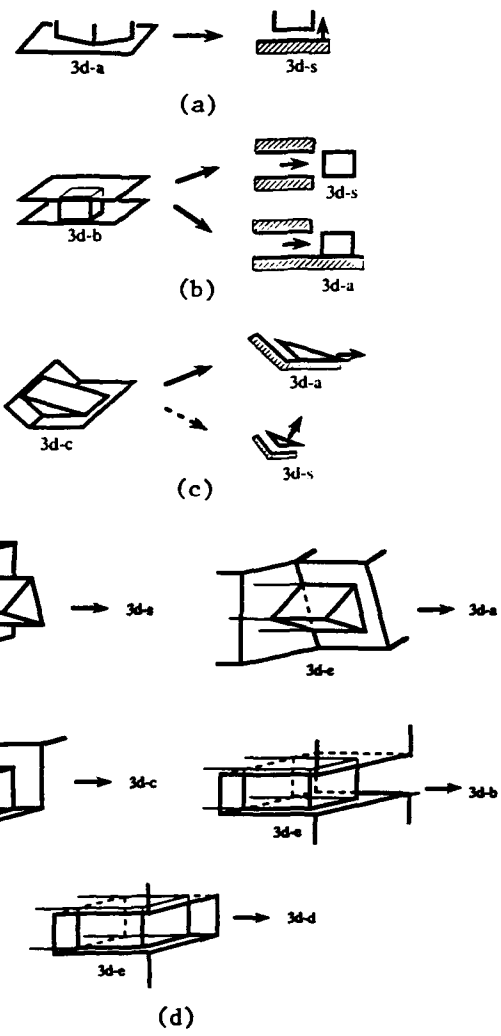


Figure 6: Examples of assembly relation transitions

By using these criteria, we will analyze each assembly relation, extract all possible assembly relation transitions, and prune unnecessary relation transitions.

We can represent relation transitions as a tree structure, as shown in Figure 7. Each node in the tree represents one particular assembly relation, and each arc represents corresponding assembly relation transitions.

3.4 Procedure tree

A procedure tree (Figure 8) is created by placing a template of manipulator operations (motion macro) at each arc separating the assembly relation nodes of Figure 7. See Table 9. The manipulator operations chosen are those which can correctly achieves an assembly relation on one node from the assembly relation on the other node.

From this analysis in Table 3, the following four motion macros are extracted:

Table 2: Possible assembly relation transitions

Relation	Possible immediate simpler relation	Transition
3d-a	A direct detach motion gives the transition from the assembly relation 3d-a to 3d-s. See Figure 6(a) for an example of the direct detach motion which causes an assembly relation transition from 3d-a to 3d-s.	s-to-a
3d-b	No direct detach motion can be applied to the assembly relation 3d-b. Lateral motions parallel to the contact faces can be applied. Depending on the shape of contact faces, it reaches either 3d-s or 3d-a. Since this variation is due to the shape of the contact face, we have to consider both cases. Figure 6(b) shows two possible relation transitions.	s-to-b a-to-b
3d-c	By applying direct detach motions, the 3d-c relation becomes either 3d-s or 3d-a. The two possibilities are not due to the shape of the contact faces; they are due to motion directions. The relation transition from 3d-c to 3d-s reduces the number of face-contacts by two, while the relation transition from 3d-c to 3d-a reduces the number by one. The latter relation transition is chosen as the desirable one by the criterion 3. Figure 6(c) shows two possible transitions due to motion directions.	a-to-c
3d-d	A direct detach motion gives the transition from the assembly relation 3d-d to 3d-b.	b-to-d
3d-e	No direct detach motions exist in case of 3d-e. Lateral motions along the axis parallel to the surrounding contact faces cause several relations, 3d-s, 3d-a, 3d-b, 3d-c and 3d-d, depending on the shape of contact faces. We have to consider these five possible relation transitions. We will refer to the axis as the *insertion axis*. See Figure 6(d) for the example.	s-to-e a-to-e b-to-e c-to-e d-to-e
3d-f	The assembly relation 3d-f becomes to 3d-s, 3d-a and 3d-c by a detach motion depending on motion directions. The relation transitions, 3d-f to 3d-s, 3d-f to 3d-a, and 3d-f to 3d-c, reduces the number of face-contacts by three, two, and one, respectively. Thus, following criterion 3, the relation transition, 3d-f to 3d-c is chosen as the desirable one.	c-to-f
3d-g	Among the two possible relation transitions, 3d-g to 3d-b and 3d-g to 3d-d, The relation transition 3d-g to 3d-d is chosen by criterion 3	d-to-g
3d-h	The assembly relation 3d-h can become to 3d-e by motion along the insertion axis.	e-to-h

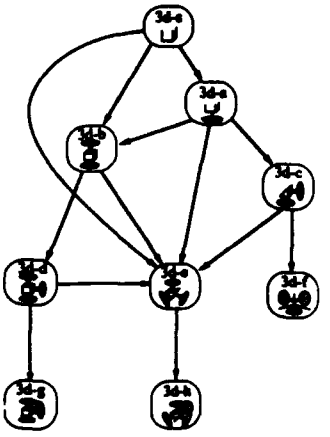


Figure 7: Assembly relation transitions represented as a tree

Table 3: Motion macro

Trans	Assigning motion macro	Macro
s-to-a	The relation transition from 3d-s to 3d-a is realized by an attach motion which contains a motion component toward the contact direction. Among several attach motions, pure motion towards the contact direction until face contact is the easiest. Thus, we assign the corresponding template of motions to the relation transition from 3d-s to 3d-a and refer to this template as *move-to-contact* motion macro.	move-to-contact
s-to-b	First we have to align the configuration of the object so that we can translate it between the two contact faces. Then, we have to translate the object parallel to the contact faces. We assign the corresponding template of motions to the relation transition from 3d-s to 3d-b and refer to this template as *insert-between* motion macro.	insert-between
a-to-b	Since the configuration of the object is already aligned, it is only necessary to translate the object parallel to the two contact faces. We assign the corresponding template of motions to the relation transition and refer to this template as *move* motion macro.	move
a-to-c	At 3d-a, one face contact is already achieved. The relation transition from 3d-a to 3d-c is realized by an attach motion along the contact face of 3d-a toward another contact face of 3d-c until two-face contact occurs. Among several such motions, pure motion perpendicular to the intersection lines between two contact faces is selected. We achieve this relation transition by using the same template of operations for the relation transition from 3d-s to 3d-a, *move-to-contact*. Thus, we assign the *move-to-contact* motion macro to the relation transition from 3d-a to 3d-c.	move-to-contact
b-to-d	The relation transition from 3d-b to 3d-d is realized by an attach motion parallel to the two opposite contact faces. Among several attach motions, as is the case in 3d-a to 3d-c, pure motion is selected. For this relation transition we use the same macro *move-to-contact*.	move-to-contact
s-to-e	First we have to align the configuration of the object so that it can be translated along the insert axis. Then, we have to translate the object along the axis. We assign the corresponding template of motions to the relation transition from 3d-s to 3d-e and refer to this template as *insert-into* macro. Note that the *insert-between* macro only aligns the object parallel to a pair of contact faces. The macro allows the rotation and translation freedom along the contact faces. On the other hand, the *insert-into* macro does not allow such freedom; it only allows the object to translate along the insert axis.	insert-into
a-to-e	The object configuration is already constrained on one contact direction in the previous 3d-a relation. Thus, by using the *insert-between* motion macro, we align the object along the insert axis and translate it into the hole.	insert-between
b-to-e	ditto	insert-between
c-to-e	The motion of the object is already constrained; the object is only movable along the insert axis. We use the *move* macro to make the relation transitions.	move
d-to-e	ditto	move
c-to-f	The relation transition from 3d-c to 3d-f is realized by an attach motion along the intersection line of the two contact faces of 3d-c until tridirectional contact occurs. For this relation transition, we use the same macro *move-to-contact*.	move-to-contact
d-to-g	At 3d-d, the object can only translate along intersection lines among contact faces. By using the *move-to-contact* macro, we achieve the tetradirectional contact.	move-to-contact
e-to-h	The relation transition from 3d-e to 3d-h is achieved by the *move-to-contact* macro along the insert axis until the tetradirectional contact occurs.	move-to-contact

- *move* - a motion sequence for this macro is realized by translating a manipulated object from the starting configuration to the ending configuration.
- *move-to-contact* - a motion sequence for this motion macro is realized by translating a manipulated object until it contacts a face of an environmental object, then fitting a manipulated object face to the contact environmental face.

  If we have precise configurations, we can achieve the contact and fitting operations by using such configurations. Otherwise, these operations require some sensory feedback to detect the occurrence of contact and fitting. See [14] for a detailed implementation of the macro as a skill in a force feedback type manipulator.
- *insert-between* - a motion sequence for this motion macro is realized by first aligning a manipulated object between a pair of contact environmental faces, and then translating it between the pair of contact faces to the ending configuration.

  If we have precise configurations, we can achieve align motion and translation motion using the configurations. Otherwise, the align motion requires some sensory feedback. See [14].
- *insert-into* - a motion sequence for this motion macro is realized by aligning a manipulated object along the insert axis, and then translating along the axis to the ending configuration.

  If we have precise configurations, we can achieve align motion and translation motion using the configurations. Otherwise, the align motion requires some sensory feedback. See [14].

Figure 8 represents a completed procedure tree.

3.5 Task models

A task model consists of an assembly relation transition, a motion macro, and the necessary parameters required to expand the motion macro into a sequence of manipulator commands. For example, Figure 9 shows the task model corresponding to the transition from 3d-s to 3d-a. The starting and end relation slots contain 3d-s and 3d-a, respectively. The action slot contains the move-to-contact motion macro. In order to achieve the motion, it is necessary to know the previous configuration and end configuration of the manipulated object. The corresponding parameters are prepared as task parameters. The values corresponding to these parameters are obtained by the task instantiation module at run time.

Thirteen task models corresponding to all arcs in the tree are prepared. They are attached to the procedure tree.

4 Implementation of APO system

How are task models used to recover human assembly tasks in the APO system? The task recognition mechanism will be explained in the following examples. The example system consists of three classes of objects, (any of which can appear in the scene): castle, block, and stick (Figure 10).

4.1 Temporal Segmentation

The system assumes that at the beginning of each assembly task human intervention occurs in the scene and at end of

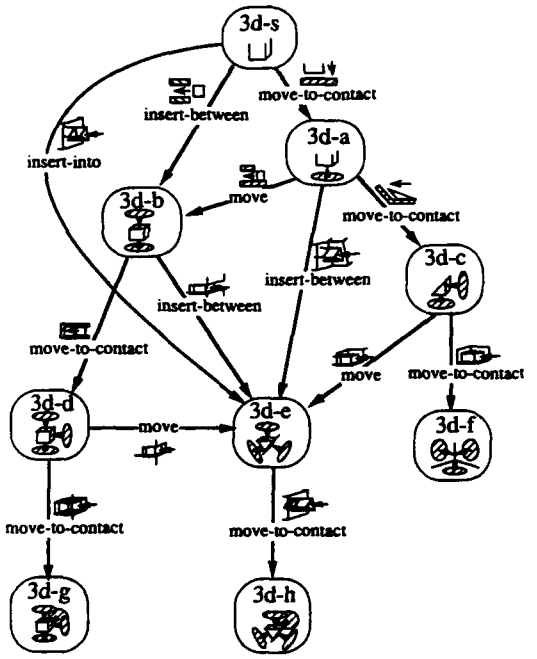


Figure 8: Procedure tree.

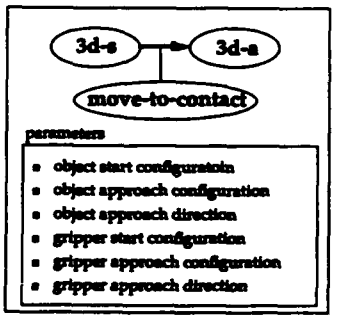


Figure 9: The task model corresponding to the assembly relation transition from 3d-s to 3d-a

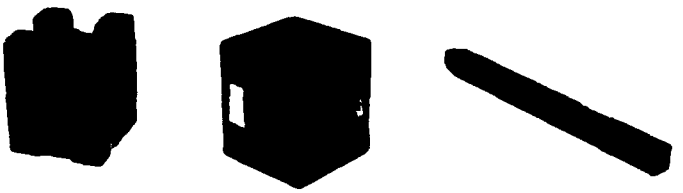


Figure 10: Castle, block and stick.

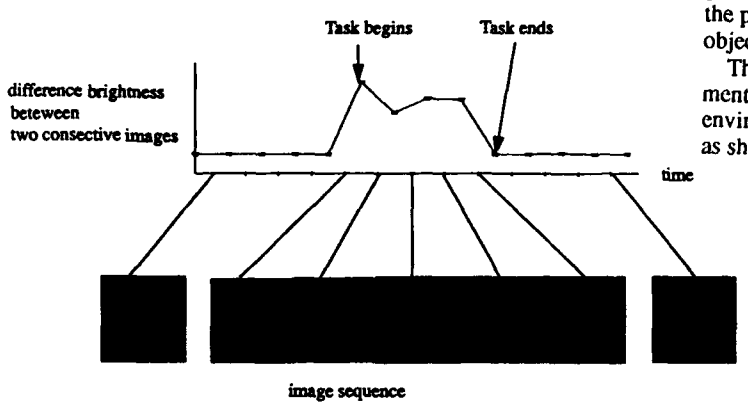


Figure 11: The system detects human intervention from the change in brightness values

the assembly task the human disappears from the scene. By using this assumption, the APO system segments a continuous image sequence given by a TV camera from the scene into a finite number of meaningful chunks.

By using the level change in the brightness difference, the system can detect human intervention. Figure 11 shows a continuous image sequence of a scene given by a TV camera, while the human operator is putting a castle on the table. Before human intervention, the scene consists of only still objects, thus the difference between two consecutive images is at the quite level. When human intervention occurs, the brightness difference is large due to the motion of human and manipulated object in the scene. This disturbance continues until the end of the assembly operation. After the human hand disappears, the scene consists of only still objects. Thus, the brightness difference returns to the quite level.

We have been using this method for detection for several live demos repeated continuously for several days, and the method never failed.

4.2 Object Recognition

Objects in the scene are recognized from range data. In our current implementation, b/w images are used only for detecting the completion of one assembly task. More reliable range data are used for analyzing the scene. After a certain period after the detection of the completion of one assembly task, the APO system invokes the range finder and measures range information in the scene. The APO system then generates a difference image between the range image from the previous step (before the assembly task) and the range image from the current step (after the assembly task).

The system applies a segmentation program to the difference image and obtains any newly appearing regions. These new regions correspond to the faces of the manipulated object by the assembly task. See Figure 12

An object recognition program is applied to the new regions. After recognizing a manipulated object from the regions by fitting a geometric model to range data on the regions, the program determines the configuration of the manipulated object.

The system maintains the configurations of other environmental objects. The system represents these manipulated and environmental objects in the Vantage geometric modeler [1] as shown in Figure 13.

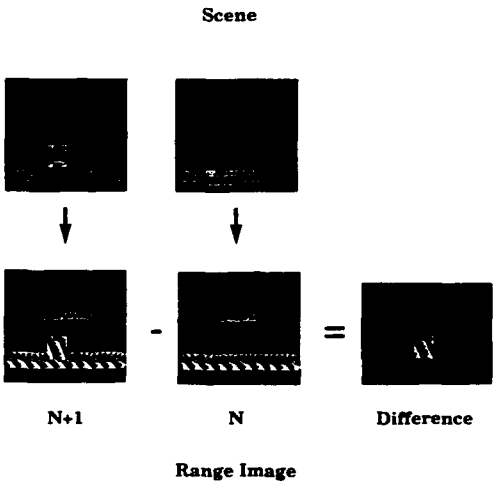


Figure 12: The difference in range data

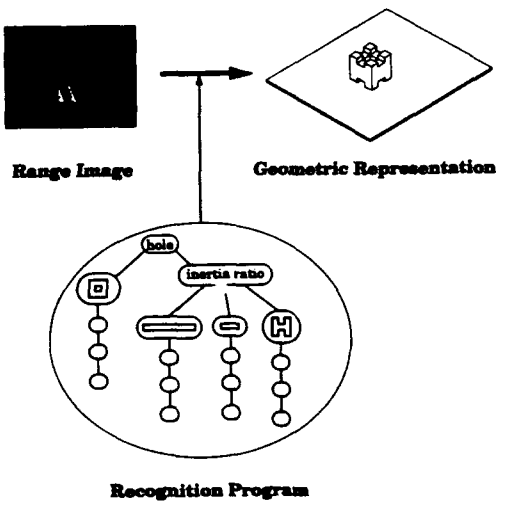


Figure 13: Object recognition: a recognition program is applied only to any newly appearing regions, and recognizes only manipulated object. The recognition results are represented by Vantage.

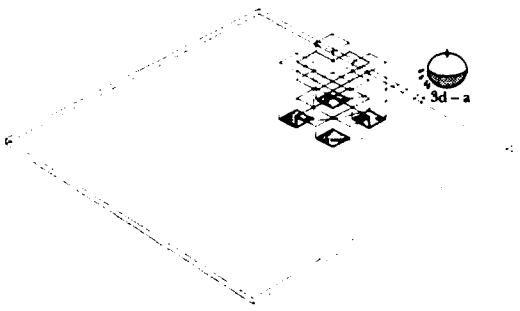


Figure 14: Extracted contact faces and assembly relation.

4.3 Task Identification

By using the transformation from body coordinate systems to face coordinate systems, (available from the Vantage geometric modeler), the configurations of the faces of the manipulated and environmental objects are obtained.

The system extracts contacting face pairs from the face configurations. Here, a contacting face pair is a face from the manipulated objects and a face from an environmental object, which have the same face equations and whose surface normals are opposite to each other.

The system determines the assembly relation based on the contacting face pairs by analyzing the contact directions of pairs. Here, the contact direction is defined as the normal direction from the environment faces to the manipulated object faces as previously defined. The contact pairs are grouped into a set of contact directional groups so that each group has face pairs with the same contact direction. By examining the occurrence of directions, we can determine which assembly relation occurs by the assembly task.

The system recognizes the contact faces and contact directions as shown in Figure 14. From the contact faces in Figure 14, the system determines that the current assembly relation is 3d-a.

Before the assembly task, the castle does not exist in the scene. Thus, before the assembly task, the assembly relation between the castle and the table was 3d-s. After performance of the assembly task, the manipulated castle established a 3d-a assembly relation with the environmental object, the table.

From this observation, the system recognizes that the assembly relation transition, 3d-s to 3d-a, occurs due to the assembly task. The corresponding task mode 3d-s to 3d-a is extracted from the corresponding arc along the procedure tree.

4.4 Task Instantiation

In this example, at the previous step, the castle was stored on the warehouse table. Thus, the assembly relation transitions during the entire assembly task are

- **3d-a to 3d-s**: detach the castle from the warehouse table to the departure configuration.
- **3d-s to 3d-s**: bring the castle from the departure configuration to the approach configuration in free space.
- **3d-s to 3d-a**: move-to-contact the castle to the working table from the approach configuration.

Thus, the corresponding three task models are instantiated: *a-to-s*, *s-to-s*, and *s-to-a*.

The following procedure is executed to instantiate a task model:

- obtain an abstract task model from the data base,
- obtain necessary parameters for the motion-macro (i.e. motion direction and translation distance) derived from the object recognition results.
- obtain the necessary motion macro (a sequence of manipulator motions) by consulting the action slot of the task model,

The instantiation of task models occurs in the reverse order, *s-to-a*, *s-to-s*, and *a-to-s*.

The *s-to-a* task model has a *move-to-contact* motion macro in the action slot. The task model examines each object model and determines grasp configurations, how to grasp the object with respect to the body coordinate system, and the specified grasping method. In the current implementation, each object model has predetermined grasping configurations. The task model chooses an appropriate grasping configuration and recalculates it based on the current body configurations. The task model determines the grasping configuration of the castle based on the observed castle configuration. The task model also determines the stack configuration of the castle on the table in a similar manner. The system then inserts these parameters to the corresponding slots in the instantiated task model.

The global motion is also implemented as a task model, *s-to-s*. This task model has a motion macro, *move*. The current implementation does not consider collision between the manipulated object and environmental objects. It assumes that space above a certain level of height is free space. The task model incorporates the path from the departure configuration to the high position, the high position to another high position above the approach configuration, and the second high position to the approach configuration. These configurations are obtained from the old and new configurations of the manipulated objects. These values are inserted into their slots in the instantiated task model.

The disassembly task is also implemented as a task model. The current implementation does not observe the warehouse table due to the field of view of the range finder. Thus, the assembly relation transition, 3d-a to 3d-s, which occurs at the warehouse table, is given to the system as a priori knowledge. The system instantiates a disassembly task model, *a-to-s*. This task model has a motion macro, *move* in the action slot. The grasp configuration for the disassembly task is obtained from the geometric model in a similar manner to the assembly task model. This value is stored in the corresponding slot in the instantiated task model.

The system finally performs the operations given by the three task models sequentially: *a-to-s*, *s-to-s*, and *s-to-a*. Figure 15 shows the final move-to-contact operation by a manipulator.

4.5 Additional examples

Figure 16(a) shows a human operation for inserting a stick in a hole of the block. The system recognizes the contact faces (Figure 16(b)). From the normal direction of contact faces, the system generates tetra directional contact. By examining

the directions of the contacts, the system determines that the observed assembly relation is 3d-e.

Currently, the vision system cannot detect intermediate relation transitions such as from 3d-b to 3d-e due to our temporal segmentation method. It can only detect the relation transition from 3d-s to 3d-e. Thus, the system explores all the possible paths in the procedure tree between 3d-s and 3d-e. Then, by examining the shape of contact pairs, the system infers which path occurs.

More precisely, the relation transition from 3d-s to 3d-e corresponds to five paths; direct path, via 3d-b, via 3d-a and 3d-b, via 3d-a and via 3d-a and 3d-c. All the arcs to the 3d-e, however, have the same assembly action (and disassembly action), translation along the axis. In Vantage, the disassembly action is applied to the current geometric representation of the manipulated and the environment objects to find the previous assembly relation. The system examines the vertex coordinates of all the contact faces, projects them to a plane parallel to the translation directions, and determines which assembly relation occurs due to this translation action. In this example, the system finds that all the boundary edge vertices on the contact faces have the same coordinate system along the translation directions. From this, it concludes that the 3d-s to 3d-e relation transition occurs.

The *s-to-e* task model has a motion macro, insert-into in the action slot. Using the predetermined grasp configuration and the observed stick position, the system performs the insert operation as shown in Figure 16(c).

Figure 17 shows other examples constructed successfully by the system.

5 Conclusion

We have described an Assembly-Plan-from-Observation (APO) system that can observe an assembly task performed by a human, recognize scene objects, relations among those objects, and actions on them, and produce corresponding operational plans for a robot. Our work will open a new domain of object recognition applications and provide a revolutionary way of programming robots.

The current system analyzes human operation and generates the fine motion plan from observation among polyhedral objects. Future directions include how to generate grasp plans and global motion plans from observation.

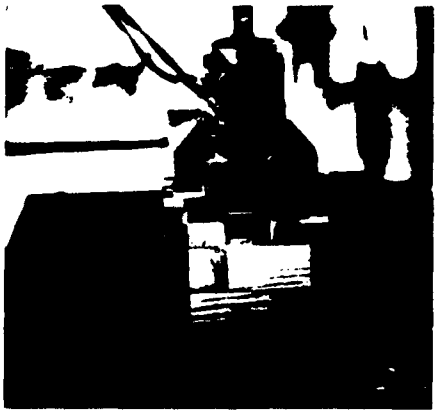


Figure 15: Put a block on the table with a manipulator.

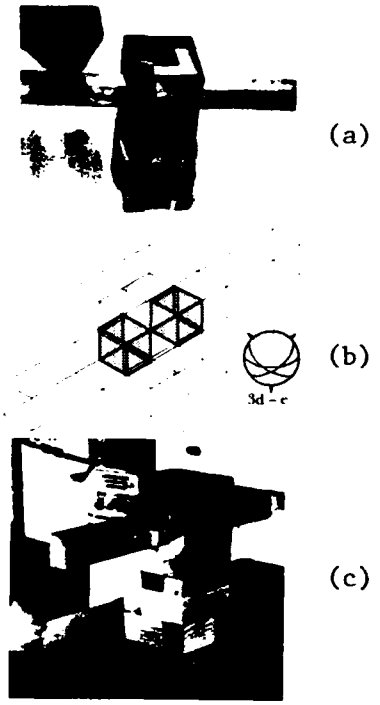


Figure 16: Insert a stick to the block; (a) input scene, (b) face contact, (c) system performance.

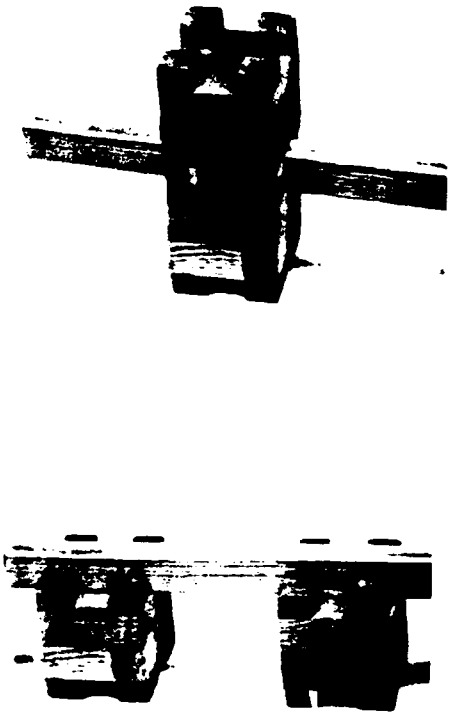


Figure 17: Additional examples.

Acknowledgement

Raj Reddy and Takeo Kanade provided useful comments and encouragements. Bradley Nelson proofread the draft of this manuscript and provided useful comments.

Takashi Suehiro was on leave of absence from Electrotechnical Laboratory, Tsukuba, Japan.

References

[1] P. Balakumar, J.C. Robert, R. Hoffman, K. Ikeuchi, and T. Kanade. Vantage: A frame-based geometric/sensor modeling system – programmer/user's manual v1.0. Technical report, Carnegie Mellon University, Robotics Institute, 1989.

[2] R. Finkel, R. Taylor, R. Bolles, R. Paul, and J. Feldman. AI, a programming system for automation. Technical Report AIM-177, Stanford University, Artificial Intelligence Laboratory, Stanford, CA, 1974.

[3] S. Hirai and Sato. T. Motion understanding for world model management of telerobot. In *IEEE Intern. Conf. on Intelligent Robots and Systems*, pages 124–131, 1989.

[4] K. Ikeuchi and K. S. Hong. Determining linear shape change: Toward automatic generation of object recognition programs. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 53(2), March 1991. a longer version, containing programs, is avaiable as CMU-CS-88-188.

[5] Y. Kuniyoshi, H. Inoue, and M. Inaba. Design and implementation of a system that generates assembly programs from visual recognition of human action sequences. In *Proc. of IEEE Intern. Workshop on Intelligent Robots and Systems*, pages 567–574, August 1990.

[6] L.I. Lieberman and M.A. Wesley. Autopass: an automatic programming system for computer controlled mechanical assembly. *IBM Journal of Res. Develop.*, 21(4):321–333, 1977.

[7] T. Lozano-Perez. Automatic planning of manipulator transfer movements. *IEEE Trans. System Man and Cybernetics*, SMC-11(10):681–689, 1981.

[8] T. Lozano-Perez, M.T. Mason, and R.H. Taylor. Automatic synthesis of fine-motion strategies for robots. In M. Brady and R. Paul, editors, *Robotics Research 1*, pages 65–96. MIT Press, Cambridge, MA, 1984.

[9] T. Lozano-Perez and P.H. Winston. Lama: a language for automatic mechanical assembly. In *Proc. of 5th Intern. Joint Conf. on Artificial Intelligence*, pages 710–716, 1977.

[10] L.S.H. Mello and A.C. Sanderson. A correct and complete algorithm for the generation of mechanical assembly sequences. In *Proc. of IEEE Intern. Conf. on Robotics and Automation*, pages 56–61, 1989.

[11] R.J. Popplestone, A.P. Ambler, and I. Bellos. An interpreter for a language for describing assemblies. *Artificial Intelligence*, 14(1):79–107, 1980.

[12] N.O. Sliwa and R.W. Will. A flexible telerobotic system for space operations. In *Proc. Space Telerobotics Workshop*, pages 285–292, Pasadena, 1987.

[13] T. Suehiro and K. Takase. Representation and control of motion in contact and its application to assembly tasks. In *Proc. of Intern. Symposuim of Robotics Reseach*, pages 367–374, Tokyo, Japan, August 1989.

[14] T. Suehiro and K. Takase. Skill based manipulation system. *Journal of the Robotics Society of Japan*, 8(5):47–58, October 1990. (in Japanese).

[15] R. Thibadeau. Artificial perception of actions. *Cognitive Science*, 10(2):117–149, 1986.

[16] S. Tsuji, A. Morizono, and S. Kuroda. Understanding a simple cartoon film by a computer vision system. In *Proc. of 5th Intern. Joint Conf. on Artificial Intelligence*, pages 609–610, 1977.

[17] D.E. Whitney. State space models of remote manipulation tasks. In *Proc. of 1st Intern. Conf. Artificial Intelligence*, pages 495–507, 1969.

[18] P.H. Winston. The mit robot. In Machine Intelligence 7, editor, *Meltzer, B. and Michie, D.M.* Edinburgh Univ. Press, Edinburgh, U.K., 1972.

[19] P.H. Winston. Learning structural descriptions from examples. In P.H. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, New York, 1975.

TRACKING, DETECTION AND 3D REPRESENTATION OF POTENTIAL OBSTACLES USING AFFINE CONSTRAINTS[1]

Harpreet S. Sawhney **Allen R. Hanson**

Computer and Information Science Department
University of Massachusetts, Amherst, MA

Abstract

In many man-made environments, obstacles in the path of a mobile robot can be characterized as *shallow*, i.e. they have relatively small extent in depth compared to the distance from the camera. We present a framework for tracking and reconstruction of shallow structures. An affine transformation is used as a dynamic model for tracking potential obstacles, and for their 3D reconstruction as shallow structures. Results of applying the constraint of *affine trackability* for automatic identification and 3D reconstruction of shallow structures in realistic scenes are presented. It is also shown how this approach can handle independent object motion, occlusions and motion discontinuity.

1 Introduction

Identification and 3D representation of potential obstacles is crucial for the success of autonomous visual navigation. Much of the work in recovering scene structure from monocular vision has concentrated on deriving depths of points or lines but has achieved only limited success. Both the motion and structure computations suffer from inherent ambiguities [2] in many realistic scenarios and also are very sensitive to noise in correspondences or flow extraction [16]. The recovery of aggregate 3D structures is generally left to some later stage in which features are grouped into objects or surfaces that could be potential obstacles.

In the approach presented here, the goal is to discover aggregate structures in the imaged scene which can be characterised as *shallow structures*. Shallow structures are 3D structures with the property that the difference in depth within the whole structure is small compared to its distance from the camera. Figure 1 shows an image of a hallway. This scene consists of compact structures like the cones and the trash can, and extended structures like the walls, the floor and the ceiling. When viewed from distances at which it might be desirable for a mobile robot to represent these internally, the variation in depth within the compact structures is small compared to their average distances from the camera. That is, the compact structures can be characterized as shallow at distances where the the robot might need an internal representation of the structures.

[1]This work was supported in part by DARPA (via TACOM) under contract number DAAE07-91-C-R035.



Figure 1: **A Hallway scene with shallow and non-shallow structures.**

Given that many potential obstacles can be characterised as shallow, the computational goal is to automatically identify shallow structures in the environment, maintain their identity or correspondence over time (tracking) and reliably reconstruct their 3D position with respect to the camera. In [14], we demonstrated how the 3D motion and structure of a shallow object in motion, relative to the camera, can be well approximated by a four-parameter affine transformation. A framework was presented for tracking shallow objects over time under the affine constraints.

Tracking under the affine constraints can also be utilised for the automatic identification of shallow structures as distinct from their background, as we demonstrate here. Furthermore, reliable reconstruction of the segmented shallow objects is also demonstrated. It is also shown how this approach can handle independent object motion, occlusions and motion discontinuity.

An advantage of the approach described here is that 3D structure information is derived reliably without the intermediate step of explicit computation of the 3D motion parameters. Recall that the well-known inherent ambiguities ([2, 17]) in the process of decomposing the image motion into a 3D rotation and a translation can lead to large errors in the 3D structure estimation.

Furthermore, the 3D location and the dynamics of the entire aggregate structure are directly represented instead of the depth of more primitive tokens like points and individual lines. The derived description of the scene can be viewed as a set of

fronto–parallel planes (*cardboard–cut–out* surfaces) of constant depth, one for each shallow object in the scene.

2 Relationship to Previous Work

In [7] and [8], a locally–constant acceleration model is used for tracking of individual 2D line segments over a sequence of frames. Williams and Hanson [18], in their work on flow–predicted line correspondences, have demonstrated that for translations in depth, reliable depth can be computed by measuring the temporal magnification (looming) of lengths and regions at approximately constant depth. Their method was demonstrated on manually selected virtual line segments and regions in the image, each of whose vertices is defined as the intersection of two lines. Automatic segmentation and temporal–persistence in tracking was not addressed. Nelson and Aloimonos [13] use the divergence of the flow field between a pair of frames to divide various regions in the image into surfaces at different depths with respect to the camera.

One of the earliest attempts at describing the scene as planar patches and its subsequent segmentation into multiple object motions was that of Adiv [1]. His approach employed the constraints on image flow from the rigid motion of a planar patch to group image regions, each region corresponding to such a motion. The input used was sparse or dense image flow and the associated confidence measures between a pair of images [4]. Since the method is based on image flow, it is not very reliable when the scene is composed primarily of textureless surfaces. Furthermore, Adiv's approach was limited to descriptions based on only two image frames and extensions to multiple frames have not been proposed. Faugeras and Lustman [9] also suggest an approach for reconstructing the scene as planar patches based on line tokens.

3 Affine Describability and Trackability

This section presents a brief review of the derivation of the affine constraint for shallow structures, and of its use in tracking. (Please refer to [14] for a detailed discussion of the affine describability and trackability constraints.)

3.1 Affine Describability

It was shown in [14] that the image projections of a shallow structure can be approximated by a four–parameter affine transformation. That is, given a 3D structure which can be well approximated by a fronto–parallel plane (shallow structure), its image projections at two closely spaced time instants are related through:

$$\frac{1}{f}p' \approx \frac{1}{f}sR_s p + t, \quad t = s\Omega_{xy} + \frac{1}{Z_0'}T_{xy} \quad (1)$$

where, p and p' are the corresponding imaged points of a shallow structure at times t and $t+1$ respectively, s is the scale defined as the ratio of average depths at the two time instants, R_s is the 2×2 rotation matrix for the rotation around the optical axis (z–axis), t is the translation in the image plane, Ω_{xy} and T_{xy} are the vectors representing the x and y components of the 3D rotational and translational vectors respectively, Z_0' is the average depth at the second time instant, and f is the focal length of the camera.

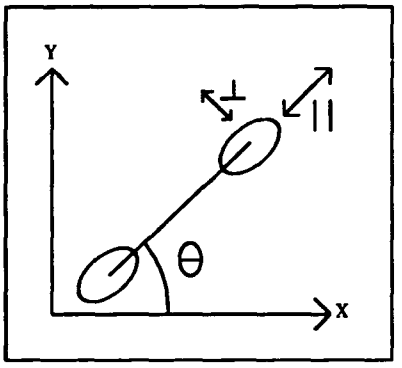


Figure 2: **Parallel and perpendicular endpoint uncertainties.**

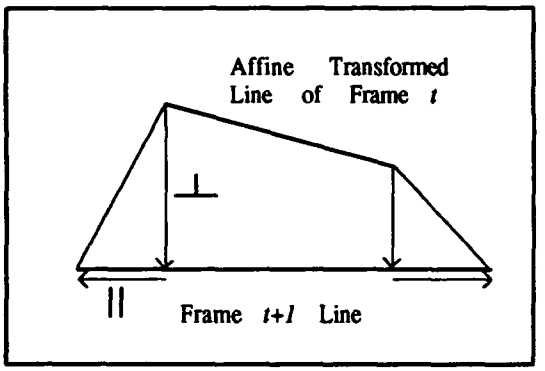


Figure 3: **The Parallel and perpendicular error components.**

3.2 Affine Parameters and their Covariances

A set of noisy line correspondences are used to compute the best affine motion parameters in the image plane. The model for noise in the extracted line segments is shown in Figure 2 [8, 14]. The uncertainties in the endpoints of a line can be modeled as variances, $\sigma_{\parallel}^2$ and $\sigma_{\perp}^2$ which are the parallel and perpendicular uncertainties, respectively, in a coordinate system aligned with the line as shown in the figure. Based on this noise model, a weighted error measure (Figure 3) is formulated to relate the image lines of a structure with the unknown affine parameters. The error measure is a sum of the parallel and perpendicular components of the vectors joining the corresponding endpoints of the line in frame $t+1$ and the affine transformed line in frame t [3]. Each of the components is weighted according to the parallel and perpendicular variances of the corresponding lines.

The error measure can be written as:

$$E_i = \sum_{j=1}^{2} w_{\perp_i}[(D_{ij}\boldsymbol{r}_s + \boldsymbol{t} - \boldsymbol{p}'_{ij}) \cdot \boldsymbol{n}'_i]^2 + w_{\parallel_i}[(D_{ij}\boldsymbol{r}_s + \boldsymbol{t} - \boldsymbol{p}'_{ij}) \cdot \boldsymbol{l}'_i]^2 \quad (2)$$

where i is the ith corresponding pair, j refers to endpoint 1 or 2, $w_{\perp_i}$ and $w_{\parallel_i}$ are the weights for the perpendicular and parallel error components, $D = \begin{bmatrix} x & -y \\ y & x \end{bmatrix}$ is the data matrix which is constructed using the endpoint $\boldsymbol{p} = [x \quad y]^T$ in frame t, vector $\boldsymbol{r}_s = [s\cos\omega_z \quad s\sin\omega_z]^T$ is the product of scale s and rotation, ω_z, around the optical axis, and $\boldsymbol{n}'_i$ and $\boldsymbol{l}'_i$ are the unit normal and direction, respectively, of the line in frame $t+1$. It is clear from Figure 3 that the first term in the above equation is the weighted perpendicular distance between the affine transformed endpoint of a line at t to the corresponding line in the next frame. The second term is the weighted longitudinal distance. The weights associated with each of the error components can be chosen appropriately for both points and lines extracted from the image data. For example, for lines typically $w_{\perp_i}$ is much larger than $w_{\parallel_i}$, reflecting the known noise characteristics of most line extraction algorithms.

For a set of line correspondences, the unknown parameters $\boldsymbol{r}_s$ and $\boldsymbol{t}$ can be found by minimising $\sum_i E_i$. Through a series of simple algebraic manipulations it can be shown that the following linear system gives the solution:

$$M_{tot}\boldsymbol{v}_{aff} = \boldsymbol{v}_{tot} \quad (3)$$

where M_{tot} and v_{tot} are the data matrix and vector, respectively, and v_{aff} is the vector of the unknown affine parameters (for full details, see [14]).

Given the model of uncertainty of the constituent lines in a structure, the covariances of the output affine parameters can be expressed as follows [15]:

$$\Lambda_{\boldsymbol{r}_s\boldsymbol{t}} = M_{tot}^{-1} \quad (4)$$

where $\Lambda_{\boldsymbol{r}_s\boldsymbol{t}}$ is the 4×4 covariance matrix of the affine parameters $\boldsymbol{r}_s$ and $\boldsymbol{t}$.

3.3 Tracking Shallow Structures

The affine motion constraint developed in the previous section can be used in a dynamic model to predict and track shallow structures over time.

Tracking requires the following three components:

1. A dynamic model of the motion (or change of state in Kalman filtering terminology) of a structure.
2. A match measure to choose good matches for a structure in every newly acquired frame. The predictions for searching for the potential matches are provided by the dynamic model.
3. A fusion of the current estimate of the affine motion and the 3D location parameters of a structure with those obtained from the newly acquired data.

All these components have to account for errors in extracting tokens (*measurement noise*) in images and also errors in modeling the dynamics of a structure (*plant noise*). Three sources of error have to be accounted for:

1. Measurement uncertainty in the image data on which the prediction is based.
2. Departures from modeled predictions of motion, (e.g. non–uniform motion).
3. Errors in the affine description due to departures from a fronto–parallel plane for the real shallow structure.

3.3.1 Dynamic Models

Kalman filtering provides a natural framework for expressing the dynamic models which can account for these various sources of error in a single unified framework. There are two dynamic models for the problem at hand. The first model is used to predict the affine parameters between frames $t+1$ and t given the parameters at time t. This is based on a model of uniform 3D motion. Modeling noise (plant noise) is added to account for departures from non–uniformity and also to account for any motion discontinuity. This model can be expressed in its general form as:

$$A_{t+1,t} = f_1(A_t) + \eta \quad (5)$$

where η is the plant noise term. The exact function f_1 was presented in [14]. It was also shown that in most practical scenarios, the contribution of the plant noise term can be limited to the translational part of the affine parameters thus considerably simplifying the computation. However, the model will also handle modeling noise in the other parameters as well. Note that the covariances of the predicted affine parameters are a combination of the covariances of the current parameters and the plant noise. Recall that the covariances of the current parameters already include the effects of measurement noise in the image data (Equation 4).

The second dynamic model predicts the image location of a shallow structure at time $t+1$ given the predicted affine motion parameters, and the current image location at t.

$$L_{t+1} = f_2(A_{t+1,t}, L_t) + \xi \quad (6)$$

The modeling noise term ξ accounts for the third source of error mentioned above — departures from a fronto–parallel plane for the real shallow structure, and also for the measurement noise in the image measurements at $t+1$. Again, the covariances of the predicted location are a fused estimate of the covariances of the predicted affine parameters and the measurement noise in the current location estimate. Note that the function f_2 is the one shown in Equation 1.

3.3.2 Model Matching

Kalman filtering provides a basis for predictions as well as fusion of uncertain information over time through recursive estimation. However, the correspondence problem also has to be addressed for tracking. Given the noisy predictions and a number of potential matches with the associated modeling and measurement covariances, we use the Maha-

lanobis distance [12, 14] for matching the predictions with potential matches in a newly acquired frame. Note that the match measure is computed for the shallow structure as a whole and not for its constituent lines individually. The covariances of the state vector associated with the model couple the various parameters of the model as a whole. This provides an implicit figural context for disambiguous matching while accounting for modeling and measurement uncertainties.

4 Does Affine Describability imply Shallowness ?

Before presenting an algorithm for shallow structure identification based on the formulation developed above, it is instructive to compare the four and the six parameter affine transformations.

4.1 General 2D Affine Transformation

In the above formulation, we have chosen to approximate a 3D shallow structure by a fronto–parallel plane. What is the resulting description if a plane of arbitrary orientation is chosen to approximate a shallow structure ? It can be shown that the four–parameter 2D affine description generalises to a six–parameter 2D affine description where the extra parameters account for the plane skew.

In the context of shape from textons, and object recognition, Kanade and Kender [11] and Huttenlocher [10], respectively, have shown that given a 2D affine transformation between two planes, a 3D similarity transformation (up to a reflection) which relates the two planes can be recovered. In other words, the relative orientation (up to a reflection), a 2D translation (parallel to one plane) and relative distance (inverse scale) between the planes can be recovered.

In the context of motion, the affine transformation describes a transformation between image projections of the same 3D plane at two time instants under the weak perspective projection. Thus, given a general 2D affine transformation describing the motion of an image patch, the 3D rotation, translation parallel to the image plane, and the depth (up to an arbitrary scale factor) can be recovered. Note that the reconstruction of the plane can be done only as a fronto–parallel plane at the computed depth. Its absolute orientation, that is its orientation in the camera coordinate system, is not recoverable; only the relative orientation that relates the orientation of the plane at the two time instants can be recovered. This is also the rotation between the two time instants. Thus, the description of structure achieved by both the four–parameter transformation described earlier, and a six–parameter transformation is as a fronto–parallel plane.

4.2 Depth reconstruction from the 4– and 6–parameter transformations

Either of the two transformations could be used for reconstructing shallow structures. Given that the 3D features (points and lines) in a shallow structure are distributed in depth around a nominal depth, the skew parameters in the six–parameter transformation account for the foreshortening effects due to the variation in depth. The four–parameter transformation accounts for the nominal depth only and not the distribution around it. Thus, in principle, the general affine transformation is a more accurate description of the image motion of a shallow structure than the four–parameter approximation used here. However, we have found experimentally, that if Huttenlocher's (or Kanade and Kender's) solution is used to reconstruct the shallow structure, it performs systematically worse than the four–parameter approximation when rotations between frames are small and when the 3D plane has a large slant. This will be discussed in more detail in the section on experimental results. Note that in the case of motion (unlike that of object recognition), where images are acquired dynamically with close spacing in time, rotations between frames are usually small.

5 Shallow Structure Identification and Tracking Algorithm

The formulation of Section 3 on tracking within the affine constraints is embedded in an algorithm to automatically identify shallow structures in a scene. The essential idea is that if a hypothesised structure can be consistently tracked and its 3D depth over time is consistent with a shallow structure model, then the structure is identified as shallow, otherwise it is labeled non–shallow. A minimal set of three lines (a triple) is used to define a hypothesised structure as a potential shallow structure.

At startup, no information is available about the motion of a hypothesised structure. The line tracking algorithm of Williams and Hanson [18], which matches lines to their flow–based predictions, is used to generate the initial correspondences. Using the correspondences thus derived, the initial affine motion parameters and their covariances are computed (Equations 3 and 4).

After the startup phase, for every newly acquired frame, the location of the hypothesised structure is predicted using the dynamic model presented in Section 3. Around each predicted line, a window query is performed to obtain potential matches for each line of the structure. Then all possible potential match structures are compared against the predicted structure.

In the *matching phase*, the Mahalanobis distance is computed for each potential data triple against the prediction, and the best triple below a threshold is chosen as the match. This threshold depends on the model of measurement errors in lines and allowable non–uniformity in motion. If all the errors are assumed to be Gaussian, then the Mahalanobis distance has a chi–squared distribution with the appropriate degrees of freedom [5]. A threshold on this distance can be chosen by using the chi–squared value corresponding to a desired level of confidence in accepting a match.

Once an acceptable match is found, in the *update phase*, the model's new motion parameters are computed. The covariances of the current location

vector and the computed affine parameters are also recomputed. Since depth is a part of the location vector, it is also updated. Additionally, the variance–weighted sample mean and sample dispersion of depth are updated by incorporating the new measurement.

If an acceptable match is not found in the current frame, due to inadequacies of line grouping or because of occlusion, the current prediction is upgraded to model status and its number–of–frames–missed count is incremented. The variances of its motion parameters and those of the line segments for its potential matches in the next frame are increased and consequently, the search window for the next prediction/matching phase is expanded. This permits graceful handling of many of the primary reasons for a match not being found, such as undergrouping or overgrouping, disappearance of lines, and motion discontinuities. If a match is reacquired after a lapse in the previous frame, the motion variances and the window size are reduced, but not below the levels set at the start of tracking.

The tracking phase discussed above is repeated for every newly hypothesized shallow aggregate structure for a window of frames. If 1) it has been tracked for more than half the frames in the window, and 2) its depth dispersion is within an allowed limit, and 3) its residual affine description error for all matched frames is less than a threshold, then it is declared as a shallow structure, else it is not and is dropped from further consideration.

5.1 Application of the Algorithm

The algorithm outlined above can be applied to image data in either a query mode or in an automatic mode. In the query mode, a set of lines is presented to the algorithm as a hypothesized shallow structure. The algorithm tracks the structure as described above and declares it shallow or otherwise.

In the automatic mode, triples of lines all over the image are instantiated as hypothesized aggregate structures and the algorithm automatically cycles through them and labels any given structure as shallow or non–shallow. We employ proximity and convexity as generic heuristics to create triples of line tokens as aggregate hypotheses.

Given a set of lines constituting a hypothesized shallow structure in frame 1, the tracking algorithm is applied. The tracking is done for a few frames before the structure is labeled.

6 Experimental Results

6.1 Tracking Results

We present the tracking results on two image sequences, *cones–seq* and *room–seq*, both of which were captured with a SONY B/W AVC–D1 camera, with an effective FOV of 24 by 23 degrees mounted on a Denning robot, and digitised to 256–by–242 pixels. The camera moved into the scene with a translation magnitude measured to be 1.95 feet for the *cones–seq* and 0.39 feet for the *room–seq*. Two image frames each of the *cones–seq* and the *room–seq* are shown in Figures 4 and 5, respectively. It is emphasised that

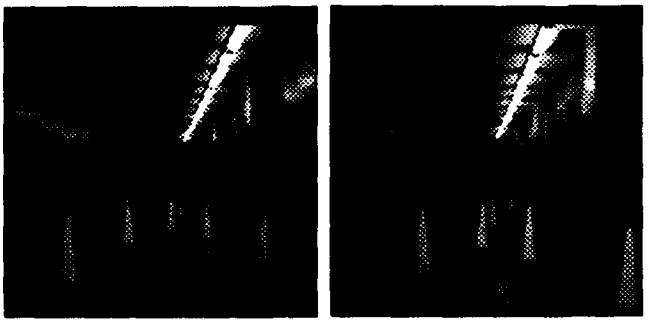


Figure 4: **Frames 1 & 6 of the *cones–seq*.**

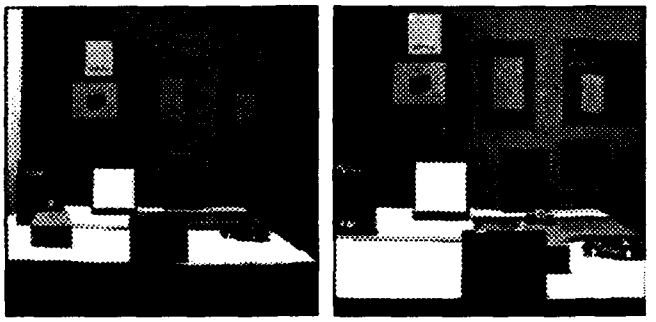


Figure 5: **Frames 1 & 8 of the *room–seq*.**

the effective motion is neither purely translational nor uniform. In each frame lines are extracted using Boldt's [6] line grouping system.

For both sequences, Figures 6–8 are to be read left–to–right and top–to–bottom. In each figure, panel a) shows the structure highlighted in bold and overlaid on lines in frame 1. Panel b) highlights the corresponding structure in frame 2; the correspondence was derived in the bootstrap phase using flow–based line tracking [18]. Subsequently, corresponding to each frame in the sequence, each panel, starting with panel c) onwards, depicts matching for each successive frame. Only the region around the structure of interest is expanded and shown in detail. The prediction windows for each line are shown as shaded areas. The central spine of these windows is the actual prediction. Thin lines show all the lines in and around the region of interest. Lines of medium thickness show the union of sets of potential matches for each line. If a match is found in a frame, it is drawn using bold lines.

Figure 6 depicts tracking of three lines on a cone. In frame 4, the left line of the cone is merged with a door line in the background by the line grouping system. No match is found for the structure in this frame but its prediction persists. In the next frame, the lines split again and the match is successfully found. This shows that the system is resilient to overgrouping errors caused by accident of viewpoints. The mechanism of model persistence allows the internal model of a shallow structure to persist even when a good match for the data is not found in a given frame. This could be due to various reasons — errors in token extraction, motion discontinuity and occlusions.

To highlight the model persistence aspect, results of tracking an independently moving object are pre-

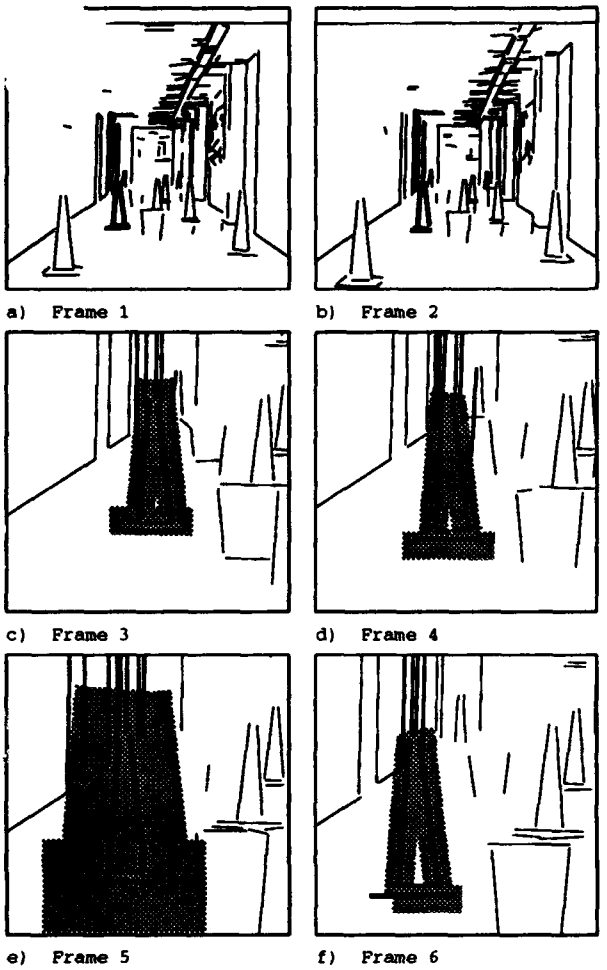


Figure 6: Tracking a cone in the Cones sequence.

sented in Figure 7. In the *room-seq*, along with the motion of the camera into the scene, an object created out of Lego blocks was moved independently. As the camera moves into the scene, this object moves from the left to the right (Figure 5). Furthermore, during the course of the motion, the Lego object gets completely occluded for three frames in the sequence and then re-emerges. Also, between frames 6 and 7, while the object is completely occluded, there is a discontinuity in the motion of the camera. The vehicle on which the camera is mounted encounters an upward sloping ramp. As a result, the predictions for the object after frame 6 go in a direction opposite to the direction of the actual motion in the image. In spite of this compounded effect of motion discontinuity and occlusion, the mechanism of model persistence and plant noise in the models enables the algorithm to maintain the object's identity and track it after it reappears.

The process is depicted in Figure 7. Note that in frame 7, when the object is still occluded, there are many potential matches within the search window but no good match is found because none of them satisfies the match criterion for the model as a whole.

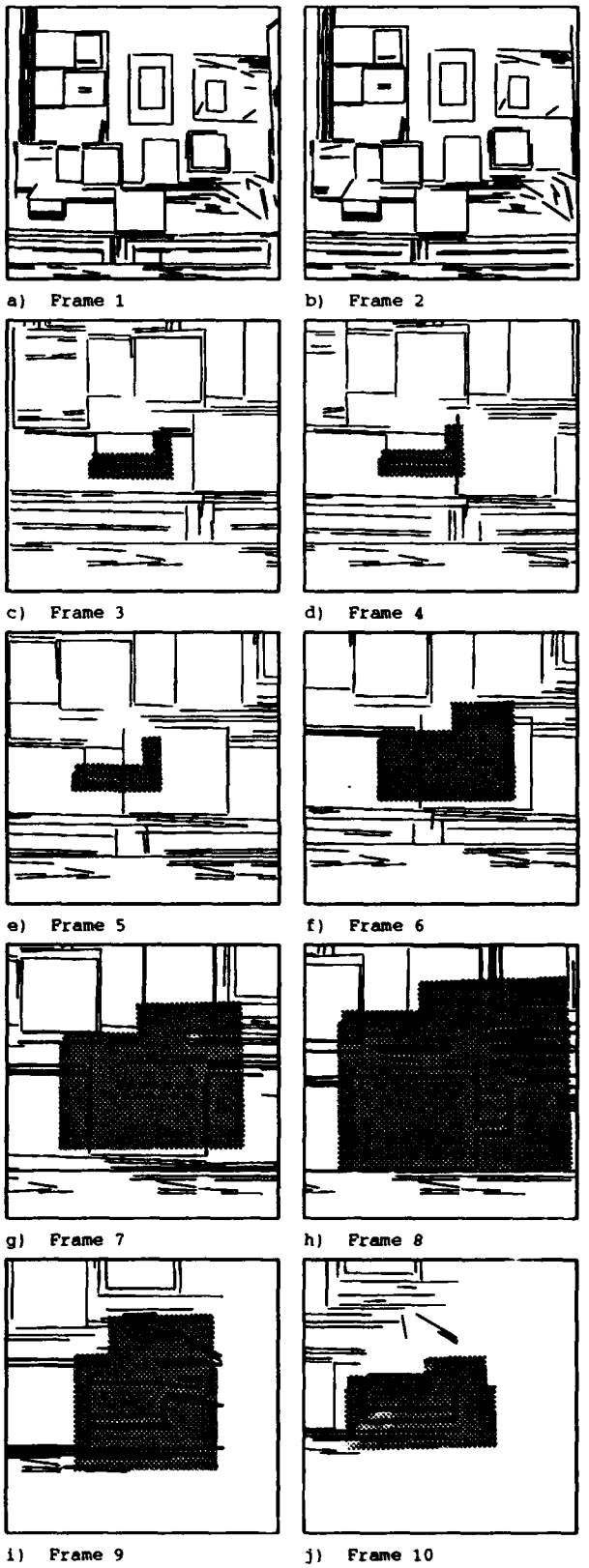


Figure 7: Tracking of a shallow independently moving object in the *room-seq*.

In contrast, if the matching was done on a per line basis, false matches could have easily been found. In frame 8, when the object reappears from behind the occluding surface, there is a lot of clutter in the search windows along with the good match. Also, notice that due to the motion discontinuity while the object was occluded, the prediction (center of the search window) has moved considerably away from the actual data. However, the model matching technique that accounts for modeling and measurement noise is able to find the object.

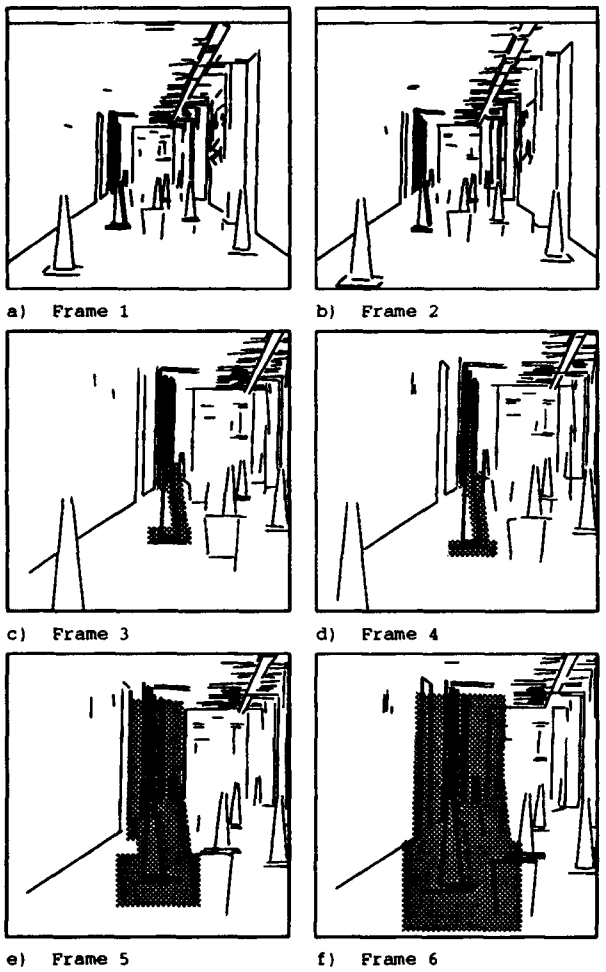


Figure 8: Non–trackability of a non–shallow triple. Two lines on a cone and one of the doorway lines in the background.

In order to use the tracking for automatic segmentation of shallow structures, it is to be shown that the tracking algorithm is unable to track non–shallow structures. With this, the affine trackability constraint can be applied to the discrimination task (shallow vs. non–shallow).

Figure 8 shows the attempt at tracking a non–shallow structure in the *cones–seq*. Two lines on a cone and one on a structure in the background have been chosen for this illustration. The figure illustrates the fact that an affine description is inadequate for describing the motion of a non–shallow structure. This is made explicit by the non–trackability of the structure over time. The structure is tracked up to frame 3 but beyond the fourth frame, the predictions deviate from the data and hence the model is lost. The deviation from shallowness is clearly expressed in the computed affine parameters in that the computed depth of the structure indicates that it is receding from rather than approaching the camera (i.e. depth is negative). The predictions in frames 5 and 6 are explicitly shown using dotted lines to contrast them with the data.



Figure 9: Shallow structures identified in the *cones–seq*.

Table 1: Computed vs. Measured Depths of some Objects in the *cones–seq* (*in feet*).

Object	Meas. Z	Comp. Z	Error (%)
Cone 1	20.0	20.24	1.2
Cone 2	25.0	25.91	3.6
Can	30.0	31.69	5.6
Cone 3	35.0	36.77	5.1
Cone 4	40.0	40.72	1.8
Cone 5	45.0	47.80	6.2
Cone 6	60.0	63.84	6.4
Door	87.1	87.70	0.7

6.2 Segmentation and Reconstruction Results

The tracking algorithm was applied to the *room–seq* and the *cones–seq* to identify the shallow structures in the scene. Line triples were automatically selected to hypothesise aggregate structures. Each of these was tested for affine trackability, resulting in its labeling as a shallow or a non–shallow structure. Figures 9 and 10 show the structures identified as shallow by the algorithm in the two sequences. In the *room–seq* and the *cones–seq*, 79 and 121 triples were found out of a total number of 180 and 167 lines, respectively.

In the *cones–seq*, amongst the two cones (cones 5 and 6) in the center of the image, two lines on cone 5 and a line on cone 6 are merged together as

Table 2: **Computed vs. Measured Depths of some Objects in the *room-seq* (*in feet*).**

Object	Meas. Z	Comp. Z	Error (%)
1	8.3	8.02	-3.4
2	13.4	12.48	6.9
3	14.57	14.6	0.2
4	18.98	18.78	-1.1
5	11.57	11.78	1.8
6	19.04	18.01	-5.4
7	20.35	19.16	-5.8
8	20.35	19.84	-2.5

a single shallow structure. This is because they are close to the FOE and are far away enough that their image motion is small. However, cone 6 is correctly labelled as a shallow structure.

The depth of some salient structures was measured with a tape measure. Tables 1 and 2 show a comparison of this ground truth with the computed depths for the *cones-seq* and the *room-seq*, respectively. The objects referred to in the tables are labelled in Figures 9 and 10.

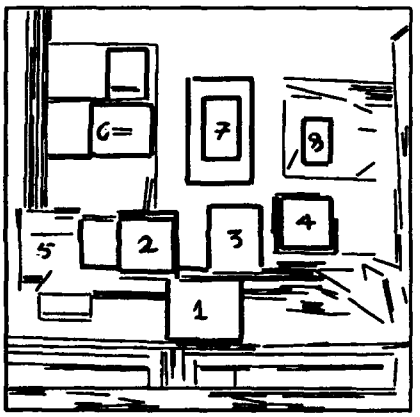


Figure 10: Shallow structures identified in the *room-seq*.

6.2.1 Comparison between the 4- and 6-parameter affine reconstruction

In Section 4, the relative merits of the 3D reconstruction of a shallow structure using the four and six-parameter affine transformations were discussed. Now we present some results of this comparison. The results are illustrated on an image sequence of a scene in which shallow planar structures of a number of orientations are present. This sequence is called the *comp-seq*; two frames are shown in Figure 11. The approximate translation-in-depth between consecutive frames is 1.4 feet.

The depths of some salient structures in the scene were measured from the camera in its position in frame 1. Recall that both the transformations reconstruct a shallow structure as a fronto-parallel plane. So, for structures which have a large slant, the ground truth depths are the average depths. Figure 12 shows some labelled objects chosen for the comparison and Table 3 shows the measured

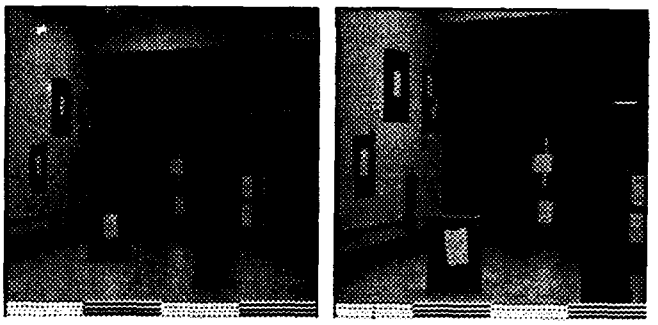


Figure 11: Frames 1 & 6 of the *comp-seq*.

depths and the depths computed by the four and six-parameter affine transformations. The depths of the highly slanted structures in the left half of the image (objects 1-3 in the table) show a larger error for the six-parameter transformation and a considerably smaller one for the four-parameter transformation. We have confirmed this with simulated data as well.

Table 3: **Depth comparisons for the four and six-parameter affine transformations in the *comp-seq* (*in feet*).**

Obj.	Meas. Z	Comp. Z			
		Affn. 4	Err. (%)	Affn. 6	Err. (%)
1	29.3	30.0	2.32	24.2	-17.42
2	31.2	31.0	-0.61	25.2	-19.31
3	33.2	34.4	3.13	25.9	-22.18
4	25.7	26.0	1.01	25.5	-0.82
5	35.8	34.2	-4.44	32.4	-9.60
6	28.2	28.3	0.39	25.9	-8.27

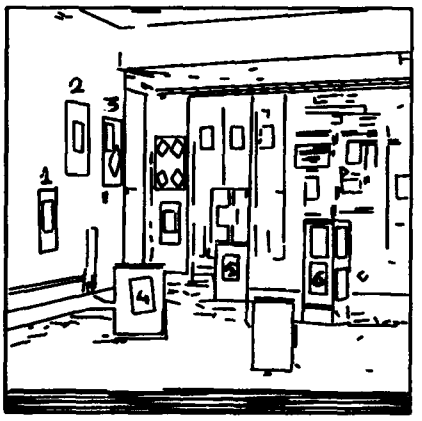


Figure 12: Labelled objects in the *comp-seq* shown in frame 1 lines.

There are two reasons for the above behavior. First, the computation of the six-parameter transformation is sensitive to the orientation of the lines chosen in the image plane. In order to understand the second reason, the equations for the 3D parameters developed in [11] can be used. The relationship between the relevant 3D parameters and the affine

transformation can be written as:

$$\frac{s^2}{det(A)} = \frac{\cos\sigma_2}{\cos\sigma_1} \quad (7)$$

where s is the scale parameter which equals the ratio of average depths of the shallow structure at instants 1 and 2, A is the 2×2 matrix for the affine transformation, and σ_1 and σ_2 are the slants of the planar approximation for the shallow structure at the two time instants. For the case of motion with small rotations, the difference in the slants, say $\delta\sigma$, is small, that is, $\sigma_2 \approx \sigma_1 + \delta\sigma$. With this approximation, the above equation can be written as:

$$s^2 = (1 - \delta\sigma \tan\sigma_1)\, det(A) \quad (8)$$

$\delta\sigma$ is related to the relative orientation or rotation between the planar positions at the two time instants. The error in s for a small error in $\delta\sigma$ can be expressed as:

$$d(s^2) = -\tan\sigma_1\, det(A)\, d(\delta\sigma) \quad (9)$$

Recall that for the six-parameter affine transformation, only $\delta\sigma$ and not σ_1 (or σ_2) can be computed. Equation 9 shows that a small error in computing $\delta\sigma$ gets magnified by the factor $\tan\sigma_1$. This factor varies from 0 to ∞ as $sigma_1$ goes from 0 to 90 degrees. Thus, for higher slants the error in scale and in the corresponding depth is higher.

7 Conclusions

In this paper, we have presented a framework for the integration of spatial constraints on generic object structure and temporal constraints on smooth motion to achieve a semantically useful description of a scene from a sequence of images. The motion of shallow structures in the image plane can be described by an affine transformation. Instead of clustering image features, observed over two frames, into an object hypothesis that is consistent with a shallow structure interpretation, the temporal evolution of a hypothesized structure is used to verify its consistency within the constraints of a shallow structure. Temporal evolution is characterized by the trackability of a structure under the affine constraint. Thus, a scene can be divided into shallow structures and the rest by the use of tracking as a verification process. Further, 3D reconstruction for the identified shallow structures can be done without an explicit computation of the 3D motion parameters.

References

[1] G. Adiv. Determining 3d motion and structure from optical flows generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):384–401, 1985.

[2] G. Adiv. Inherent ambiguities in recovering 3d information from a noisy flow field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5), 1989.

[3] Gilad Adiv and Edward Riseman. Recovery of 3d motion and structure from image correspondences using a directional confidence measure. Technical Report COINS TR 88–105, University of Massachusetts at Amherst, MA, 1988.

[4] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, 1989.

[5] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*. Academic Press, 1988.

[6] M. Boldt, R.Weiss, and E. Riseman. Token-based extraction of straight lines. *IEEE Transactions on Systems Man and Cybernetics*, 19(6):1581–1594, 1989.

[7] J. L. Crowley, P. Stelmaszyk, and C. Discours. Measuring image flow by tracking edge-lines. In *Proc. 2nd Intl. Conf. on Computer Vision*, pages 658–664, 1988.

[8] R. Deriche and O. Faugeras. Tracking line segments. In *Proc. 1st European Conference on Computer Vision*, pages 259–268, 1990.

[9] O. D. Faugeras and F. Lustman. Let us suppose the world is piece-wise planar. In *Proc. The Third International Symposium on Robotics Research*, 1987.

[10] Daniel P. Huttenlocher and Shimon Ullman. Recognising solid objects by alignment. In *Proc. Computer Vision and Pattern Recognition Conference*, pages 1114–1124, 1989.

[11] T. Kanade and J. R. Kender. Mapping image properties into shape constraints: Skewed symmetry, affine-transformable patterns, and the shape-from-texture paradigm. In J. Beck et al, editor, *Human and Machine Vision*, pages 237–257. Academic Press, NY, 1983.

[12] P. C. Mahalanobis. On the generalised distance in statistics. *Proceedings of the National Institute of Science, India*, 12:49–55, 1936.

[13] R. C. Nelson and J. Aloimonos. Towards qualitative vision: Using flow field divergence for obstacle avoidance in visual navigation. In *Proc. 2nd Intl. Conf. on Computer Vision*, 1988.

[14] H. S. Sawhney and A. R. Hanson. Identification and 3d description of 'shallow' environmental structure in a sequence of images. In *Proc. Computer Vision and Pattern Recognition Conference*, pages 179–186, 1991.

[15] Gilbert Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, MA, 1986.

[16] R. Y. Tsai and T. S. Huang. Uniqueness and estimation of 3d motion parameters and surface structures of rigid objects. In Whitman Richards and Shimon Ullman, editors, *Image Understanding 1984*, pages 135–171. 1984.

[17] J. Weng, T. S. Huang, and N. Ahuja. Motion and structure from two perspective views: Algorithms, error analysis, and error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):451–476, 1989.

[18] L. R. Williams and A. R. Hanson. Translating optical flow into token matches and depth from looming. In *Proc. 2nd Intl. Conf. on Computer Vision*, pages 441–448, 1988.

Trajectory Filtering and Prediction for Automated Tracking and Grasping of a Moving Object

Peter K. Allen
Aleksandar Timcenko
Billibon Yoshimi
Paul Michelman

Department of Computer Science
Columbia University
New York, NY 10027*

Abstract

Most robotic grasping tasks assume a stationary or fixed object. In this paper, we explore the requirements for grasping a moving object. This task requires proper coordination between at least 3 separate subsystems: real-time vision sensing, trajectory-planning/arm-control, and grasp planning. As with humans, our system first visually tracks the object's 3-D position. Because the object is in motion, this must be done in real-time to coordinate the motion of the robotic arm as it tracks the object. The vision system is used to feed a an arm control algorithm that plans a trajectory. The arm control algorithm is implemented in two steps: 1) filtering and prediction, and 2) kinematic transformation computation. Once the trajectory of the object is tracked, the hand must intercept the object to actually grasp it. We present experimental results in which a moving model train is tracked, stably grasped, and picked up by the system.

1 INTRODUCTION

The focus of our work is to achieve a high level of interaction between a real-time vision system that is capable of tracking moving objects in 3-D and a robot arm that contains a dexterous hand that can be used to intercept, grasp and pick up a moving object. We are interested in exploring the interplay of hand-eye coordination for dynamic grasping tasks such as grasping of parts on a moving conveyor system, assembly of articulated parts or for grasping from a mobile robotic system. Coordination between an organism's sensing modalities and motor control system is a hallmark of intelligent behavior, and we are pursuing the goal of building an integrated sensing and actuation system that can operate in dynamic as opposed to static environments. The algorithms we have developed that relate sensing to actuation are quite general and applicable to a variety of complex robotic tasks that require visual feedback for arm and hand control.

The system we have built addresses three distinct problems in robotic hand-eye coordination for grasping moving objects: fast computation of 3-D motion parameters from vision, predictive control of a moving robotic arm to track a moving object, and grasp planning. The system is able to operate at approximately human arm movement rates, using visual feedback to track, stably grasp, and pickup a moving object.

The system consists of two fixed cameras that can image a scene containing a moving object (see Figure 1). A PUMA-560 with a parallel jaw gripper attached is used to track the object with the of stably grasping and picking up the object as it moves. The system operates as fol-

*This work was supported in part by DARPA contract N00039-84-C-0165, NSF grants DMC-86-05065, DCI-86-08845, CCR-86-12709, IRI-86-57151, IRI-88-1319, North American Philips Laboratories, Siemens Corporation and Rockwell Inc.

lows:

1. The imaging system performs a stereoscopic optic-flow calculation at each pixel in the image. From these optic-flow fields, a motion energy profile is obtained that forms the basis for a triangulation that can recover the 3-D position of a moving object at video rates.

2. The 3-D position of the moving object computed by step 1 is initially smoothed to remove sensor noise, and a non-linear filter is used to recover the correct trajectory parameters which can be used for forward prediction, and the updated position is sent to the trajectory-planner/arm-control system.

3. The trajectory planner updates the joint level servos of the arm via kinematic transform equations. An additional fixed gain filter is used to provide servo-level control in case of missed or delayed communication from the vision and filtering system.

4. Once tracking is stable, the system commands the arm to intercept the moving object and the hand is used to stably grasp the object and pick it up.

The following sections of the paper describe each of these subsystems in detail along with experimental results.

2 PREVIOUS WORK

Previous efforts in the areas of motion tracking and real-time control are too numerous to exhaustively list here. We instead list some notable efforts that have inspired us or use similar approaches. Burt et al. [1] has focused on high speed feature detection and hierarchical scaling of images in order to meet the real-time demands of surveillance and other robotic applications. Related work has been reported by Lee and Wohn [2] and Wiklund and Granlund [3] who use image differencing methods to track motion. Corke, Paul and Wohn [4] report a feature based tracking method that uses special purpose hardware to drive a servo-controller of an arm-mounted camera. Goldenberg et al.[5] have developed a method that uses temporal filtering with similar hardware to our own. Luo, Mullen and Wessel [6] report a real-time implementation of motion tracking in 1-D based on Horn and Schunk's method. Verghese et al. [7] report real-time, short-range visual tracking of objects using a pipelined system similar to our own. Safadi [8] uses a tracking filter similar to our own and a pyramid based vision system, but few results are reported with this system. Rao and Durrant-Whyte [9] have implemented a Kalman filter based de-centralized tracking system that tracks moving objects with multiple cameras. Miller [10] has integrated a camera and arm for a tracking task where the emphasis is on learning kinematic and control parameters of the system. Weiss et al. [11] also use visual feedback to develop control laws for manipulation. Brown [12] has implemented a gaze control system that links a robotic "head" containing binocular cameras with a servo controller that allows one to maintain a fixed gaze on a moving object. Clark and Ferrier [13] also have implemented a gaze control system for a mobile robot. A variation of the tracking problems is the case of moving cameras. Some of the papers addressing this interesting problem are [14, 15, 16].

The majority of literature on the control problems encountered in motion tracking experiments is concerned with the problem of generating smooth, up-to-date trajectories from noisy and delayed outputs from different vision algorithms. Our previous work [17] coped with that problem in a similar way as in [18], using an $\alpha - \beta - \gamma$ filter, which is a form of a steady-state Kalman filter. A similar approach can be found in papers by [19, 20, 21]. In [19] a sophisticated control scheme is described which combines a Kalman filter's estimation and filtering power with an optimal (LQG) controller which computes the robot's motion. The authors have presented good tracking results, as well as stated that the controller is robust enough so the use of more complex (time-varying LQG) methods is not justified. The choice of gain matrices in the cost function and the best set of noise variances is done empirically. Paper [20] addresses the problem of uncertainty of cameras in the robot's coordinate frame. The fact that cameras have to be strictly fixed in robot's frame might be quite annoying since each time they are (most often incidentally) displaced, one has to undertake a tedious job of their recalibration. Again, the estimation of moving object's position and orienta-

tion is done in the Cartesian space and a simple error model is assumed. The paper [21] adopts 3rd order Kalman filter in order to allow a robotic system (consisting of two degrees of freedom) to play the labyrinth game.

A somewhat different approach has been explored in papers [22, 23, 24]. The auto-regressive (AR) and auto-regressive moving-average with exogenous input (ARMAX) models are investigated. It is noteworthy to point out, as stated in [22], that this is more of an implementation than a conceptual difference from the classical Kalman-filter approach since the coefficients of polynomials in ARMAX model depend on the Kalman gains.

3 VISION SYSTEM

The vision system used in this research is described in detail in [17] and we briefly review the method here. In a visual tracking problem, motion in the imaging system has to be translated into 3-D scene motion. Our approach is to initially compute local optic-flow fields that measure image velocity at each pixel in the image. A variety of techniques for computing optic-flow fields have been used with varying results including matching based techniques [25, 26, 27] gradient based techniques [28, 29, 30] and spatio-temporal energy methods [31, 32]. Optic-flow was chosen as the primitive upon which to base the tracking algorithm since it can be extracted quickly and reliably from our images, and it quantifies actual motion in the scene which we need to detect. We are using 2 fixed cameras that are calibrated with the 3-D scene, but there is no explicit need to use registered (i.e scan-line coherence) cameras. The identical algorithm for extracting optic-flow is run on each camera's image in parallel using the PIPE parallel image processor [33]. Once the motion centroids are known for each camera, they are back-projected into the scene using the camera calibration matrices and triangulated to find the actual 3-D location of the movement. This 3-D position is computed every 1/60th second, but with a processing delay of roughly 10 msec.

4 ARM CONTROL

The second part of the system is the arm control. The robotic arm has to be controlled in real-time to follow the motion of the object, using the output of the vision system. The raw vision system output is not sufficient as a control parameter since its output is both noisy as well as delayed in time. The control system needs to do the following:

- Filter out the noise with a digital filter
- Predict the position to cope with delays introduced by both vision subsystem and the digital filter
- Perform the kinematic transformations which will map the desired manipulator's tip position from a Cartesian coordinate frame into joint coordinates, and actually perform the movement

Our vision algorithm provides in each sampling instant a position in 3D space as a triplet of Cartesian coordinates (x, y, z). The task of the control algorithm is to smooth and predict ahead the trajectory, thus positioning the robot where the object is during its motion.

A well known and useful solution is the Kalman filter approach, because it successfully performs both smoothing and prediction. However, the assumption the Kalman filter makes is that the noise applied to the system is white. That fact directly depends on the parametrization of the trajectory and, unfortunately in our case, the simplest possible parametrization - Cartesian- does not support this noise model. Our previous work [17] used a variant of this approach and obtained tracking that was smooth but not accurate enough to allow actual grasping of the moving object. Our solution to this problem was to appeal to a local coordinate system that was able to model the motion and system noise characteristics more accurately, thus producing a more accurate control algorithm.

4.1 The Model of the 3D Motion

The main idea in the trajectory parametrization used in this paper is to describe a point in a *local* coordinate frame, relative to the point from

the previous sampling instant, by the triplet of coordinates (s, ϕ, z) where

- s is the length of an arc between two points
- ϕ is the "bending" of the trajectory (see figure 2)
- z is the altitude difference in two consecutive points

Due to the existence of noise, all three coordinates are random variables with certain distributions. We have made the following assumptions, as a result of both reasoning about the vision algorithm and certain necessary simplifications:

- In sampling instant k our object is in point P_k
- In the next sampling instant $k+1$ the object is in P_{k+1} and the point returned by the vision algorithm is Q_{k+1}
- Q_{k+1} is normally distributed around P_{k+1}. The noise can be expressed by its two components, tangential n_t and normal n_n
- n_t and n_n are both zero-mean, with the same dispersion and mutually not correlated. Experimentally, it has been determined that their coefficient of correlation is between 0.1 and 0.2.

Under these assumptions it can be shown that the velocity v and curvature κ are:

$$v = \lim_{T \to 0} s/T \tag{1}$$

$$\kappa = \lim_{T \to 0} \tan \varphi_0 / s_0 \tag{2}$$

where $s_0 = ||P_{k+1} - P_k||$ and $\varphi_0 = \pi - \angle P_{k-1} P_k P_{k+1}$.

What are advantages of such a parametrization? The most obvious one is the simplicity of the prediction task in this framework; all we need is to multiply the velocity $v = s/T$ by the time $\tau > T$ we want to predict ahead, as well as "bending" ϕ. The next advantage is that in order to achieve an accurate prediction, we do not need a high-order model with the mostly heuristic tuning of numerous parameters. The price we have to pay is that *filtering is not straightforward*. It turns out that we cannot just apply a low-pass filter in order to recover a DC component from s, but rather we need more elaborate approach which takes into account a probabilistic distribution of s.

While this model introduces more complexity than a standard Cartesian model, we will see below that it is more effective in allowing us to accurately predict and smooth our trajectory. The initial experiments with this model separates 3-D space into an XY plane and the Z axis, and addresses these two components of motion separately. However, the method for the XY plane can be extended to include another parameter which will create a full Frenet Frame at each instant of time in the trajectory. Our initial experiments (described below) tracked a planar curve, allowing us to use this simplification. Motion in the Z direction is tracked with a Cartesian displacement as outlined in [17].

Our model assumes the following coordinate transformation that relates the moving object's coordinate frame at one instant with the next instant in time:

$$\mathsf{Rot}(z, \phi_0) \circ \mathit{Trans}(x, s) \circ \mathsf{Trans}(z, \Delta z) \tag{3}$$

where Rot and Trans are rotation / translation around / along given axis. Presented as a 4×4 matrix, transformation (3) is

$$T_{\mathsf{delta}} = \begin{bmatrix} \cos\phi_0 & -\sin\phi_0 & 0 & s\cos\phi_0 \\ \sin\phi_0 & \cos\phi_0 & 0 & s\sin\phi_0 \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.2 Probability Distributions of s and ϕ

In this section, we will motivate the choice of model used to recover the parameter values s_0 and φ given the estimate of the arclength s. Let $s = ||Q_{k+1} - P_k||$ be the distance between the object and the next position returned by the vision algorithm. According to figure 2 we have

$$\begin{aligned} s &= \left\| \begin{bmatrix} \cos\varphi_0 & \sin\varphi_0 \\ -\sin\varphi_0 & \cos\varphi_0 \end{bmatrix} \begin{bmatrix} n_t \\ n_n \end{bmatrix} + \begin{bmatrix} s_0 \\ 0 \end{bmatrix} \right\| \\ &= \sqrt{(n_t' + s_0)^2 + n_n'^2} \end{aligned} \tag{4}$$

where n_t' and n_n' are Gaussian with dispersion σ. According to the definition of the probability

distribution, we can write the distribution $F(s)$ as

$$F(s) = \int\int_D \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\left[\left(\frac{t-s_0}{\sigma}\right)^2+\left(\frac{n}{\sigma}\right)^2\right]} dt dn \quad (5)$$

where D is a disk of the radius s.

Now by introducing substitution $t = r\cos\theta$, $n = r\sin\theta$ we get

$$F(s) = \frac{1}{2\pi\sigma^2} \int_0^s r \int_0^{2\pi} e^{-\frac{1}{2}\left[\left(\frac{r\cos\theta-s_0}{\sigma}\right)^2+\left(\frac{r\sin\theta}{\sigma}\right)^2\right]} d\theta dr$$

Distribution density is given as $f(s) = \frac{dF(s)}{ds}$ or after differentiation

$$f(s) = \frac{se^{-\frac{s^2+s_0^2}{2}}}{2\pi\sigma^2} \int_0^{2\pi} e^{-\frac{ss_0}{\sigma^2}\cos\theta} d\theta \quad (6)$$

The last integral can be expressed by a modified Bessel function $I_0(z)$:

$$f(s) = \frac{s}{\sigma^2} e^{-\frac{s^2+s_0^2}{2\sigma^2}} I_0\left(\frac{ss_0}{\sigma^2}\right) \quad (7)$$

A graph of $f(s)$ is given in figure 3. Here s_0 is fixed to 1 and σ varies from 0.4 to 1.0. Our job is to recover s_0 given $f(s)$.

It is apparent from the figure 3 that the peak value of $f(s)$ depends on σ, and drifts towards higher values as σ grows. The expectation for s also depends on σ. In particular, we have

$$s_1 = E(s) = \int_0^\infty s f(s) ds = \sigma u\left(\frac{s_0}{\sigma}\right) \quad (8)$$

where

$$u(x) = \sqrt{\frac{\pi}{2}} e^{-x^2/4}\left(I_0\left(\frac{x^2}{4}\right)+\frac{x^2}{2}\left(I_0\left(\frac{x^2}{4}\right)+I_1\left(\frac{x^2}{4}\right)\right)\right) \quad (9)$$

Here σ is the constant for the given system and it is related to s_0. In order to estimate σ we will use second-order moment:

$$s_2^2 = E(s^2) = \int_0^\infty s^2 f(s) ds = s_0^2 + 2\sigma^2 \quad (10)$$

Now by eliminating s_0 from (8) and (10) we have

$$1 = z\, u\left(\frac{\sqrt{p^2 - 2z^2}}{z}\right) \quad (11)$$

where $p = s_2/s_1$ and $z = \sigma/s_1$. Now by setting $x = \frac{\sqrt{p^2-2z^2}}{z}$ we end up with an equation

$$u_1(x) = \frac{\sqrt{x^2+2}}{u(x)} = p \quad (12)$$

Equation 12 relates our known control inputs ($p = s_2/s_1$) to x. We can create a table of values for this function offline, and then by interpolation calculate a value of x given p.

Let $x_0(p)$ be the solution of (12). Now we can express s_0 and σ as functions of s_1 and s_2 as follows:

$$s_0 = s_2 \frac{x_0\left(\frac{s_2}{s_1}\right)}{\sqrt{2 + x_0\left(\frac{s_2}{s_1}\right)^2}} \quad (13)$$

$$\sigma = s_2 \frac{1}{\sqrt{2 + x_0\left(\frac{s_2}{s_1}\right)^2}} \quad (14)$$

This method requires little on line computation - an interpolation table of values of u_1 is all we need to recover the arclength parameter s_0. Figure 5 is the experimentally measured density of s_1 taken from the triangulated optic-flow fields. This distribution's resemblance to figure 3 (the theoretical density) is clear.

To find the bending parameter ϕ_0, we use the same technique as for the distribution of s, and we get the following formula:

$$f(\phi) = \frac{\cos(\phi - \phi_0)}{\sqrt{2\pi}k} e^{-\frac{\sin^2(\phi-\phi_0)}{2k^2}}$$

where $k = \sigma/s_0$ and $\phi - \phi_0 \in (-\pi/2, \pi/2)$. It is obvious that f is symmetric around ϕ_0, which also means that the expectation $E\phi = \phi_0$. Hence, we so not need to perform a non-linear filtering to recover ϕ_0.

The graph of f for $k = 0.1$ to 0.9 and $\phi_0 = 0$ is given in figure 6.

4.3 Smoothing of the Control Inputs

In the previous section, we showed how to extract parameters s_0 and ϕ_0 from the updated positions determined from the vision system. The signals s_1, s_2^2 described in equations 8 and 10 are in fact

the smoothed versions expectations of the control signals s, s^2 which are the arclength and the arclength squared. The smoothing filter we use to compute these signals is a moving-average (MA) filter using a Kaiser window [34]. This filter provides the largest ratio of signal energy in the main lobe and a side lobe, which usually results in a filter of lower order. The windowing function is given by

$$w_K(n) = \frac{I_0(\beta\sqrt{1-(1-2n/M)^2})}{I_0(\beta)}$$

where I_0 is the modified zeroth-order Bessel function, β is the shape parameter which defines the width of the main lobe and M is the order of the filter. According to [34], β and M are given by

$$M \approx \frac{A-7.95}{14.36\Delta\omega}$$

and

$$\beta = \begin{cases} 0.1102(A-8.7), A \geq 50 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21), \\ 21 < A < 50 \end{cases}$$

where A is the stopband attenuation and $\Delta\omega = (\omega_r - \omega_c)/\omega_s$, ω_r is the stopband frequency, ω_c is the passband frequency and ω_s is the sampling frequency.

We have adopted $A = 30$ and $\Delta\omega = 0.05$ which results in $M = 30$. Since the frequency of the vision algorithm is about 60 Hz, the overall length of the window is about 0.5 seconds. We also apply this MA filter to the bending parameter ϕ.

The implementation of MA filter is straightforward: once the weights are computed off-line, a window of length M of measurements is retained and each sample is multiplied by an appropriate weight in the sampling period, which requires M multiplications and $M-1$ additions. This allows reasonably wide windows (even up to several hundreds entries) to be used in computing the smoothed signal.

4.4 Prediction and Synchronization

The host computer controls the initial vision processing and subsequent computation of control parameters described above. The host computer is able to predict ahead the trajectory using the derivation of velocity and curvature in equations (1) and (2). These updated predictions are sent to the trajectory generator that is actually controlling the robot arm. The trajectory generator is a separate system that has two parallel tasks: a low-priority task which reads the serial line receiving updated control signals and high-priority task which calculates the transformation equation and moves the manipulator. Those two tasks communicate via shared memory. The job of the robot controlling program is to synchronize its two tasks (i.e. to obtain mutual exclusion in accessing shared data), to unpack input packets read from the serial line, and to update the joint servos every 30 msec.

The asynchronous nature of the communication between the host computer and the trajectory generator can result in missed or delayed communications between the two systems. Since the updating of the robotic arm parameters needs to be done at very tightly specified servo rates (30 msec), it is imperative that the trajectory generator can provide updated control parameters at these rates, regardless of whether it has received a new control input from the host. Therefore, we have implemented a fixed gain $\alpha-\beta-\gamma$ filter as part of the trajectory generator [18]. This filter provides a small amount of prediction to the trajectory parameters if the control signals from the host are delayed.

We are using RCCL [35] to control the robotic arm (a PUMA 560). RCCL (Robot Control C Language) allows the use of C programming constructs to control the robot as well as defining transformation equations (as described in [36]). The transformation equations permit dynamic updating of arm position by generating the 4×4 transform of the moving object's position from the vision system and sending this information to the arm control algorithm.

5 GRASPING

The remaining part of our system is the interception and grasping of the object. We have examined the human psychological literature in order to find useful paradigms for robotic visual-motor coordination strategies that include arm move-

ment and grasping from visual inputs. In this section we briefly describe some relevant theories and their relation to our own work.

There are several theories on the organization of skilled human motor control. Richard Schmidt [37] has proposed a theory of generalized motor programs, or movement *schemas*. In this view, a skilled action is composed of an ordered set of parametrized motor control programs of short duration (less than 200 msec), each of which accomplishes one part of the task. As one program is completed, the next one is executed. Generalized motor programs accomplish several objectives: (1) they specify *which muscle* to move in a given motion; (2) the *order of contraction* of the muscles; (3) the *phasing* within the sequence, i.e., the temporal relationships among the contractions; (4) the *relative force* of each element. At the initiation of a skilled task, the parameters of the motor control program are determined by sensory input and task demands, and then the programs are executed to completion. If the wrong program is selected for some reason, the program cannot be stopped by use of sensory information. Similarly, in playing table tennis, the motion of the racket is determined before the beginning of the swing and visual input has little effect after the initiation of motion. As an example of Schmidt's theory, the skilled task of grasping a moving object could be partitioned into two motor control schemas: one to position the arm and a second one to control the grasping action.

The schema concept maps into Von Hofsten's ideas about the development of grasping skills in children [38] He believes there are two separate sensorimotor systems responsible for reaching: one for approaching the target and one for grasping it. During early childhood, the precise timing between these two systems develops as the child learns how to catch. The reaching system develops first, before a child is capable of grasping. But even before he is capable of closing his hand at precisely the right moment, he has begun to develop the ability to move his hand toward a moving object and predict the location at which his hand will intercept the object. With growth, a child learns to control the timing between reaching and grasping, that is, to close his hand at the correct moment. Experimental evidence has shown that there is a window of approximately 14 msec during which the hand must begin closing. Unlike Schmidt, however, Von Hofsten does not consider vision and grasping to be two mutually exclusive tasks [39] Visual tracking is used to guide the reaching arm *during* its motion, not only before motion. A coordinated motion is a combination of perceptual schemas and motor schemas (see Iberall and Arbib [40]).

Vision is used during the reaching phase of the task for what psychologists call "prospective control". Prospective control corresponds to predictive filtering, as used by control theorists. In grasping a moving object, it is necessary for the hand to move not to the current position of the object, but to plan ahead to where it will be shortly. Vision, rather than haptics, provides the basis of prospective control because touch cannot provide the anticipatory information required to predict the course of a moving object. There are two predominant theories about what visual schema is used to track a moving object and aid in predicting the intersection of the reaching hand and that object. Lee [41] proposes the use of vision to measure the expansion of the image on the retina in order to estimate the time until contact. The attraction of this theory is that humans would not need to compute the velocity and location of the moving object, but would calculate the more useful time-until-contact information. A person catching an object uses this image to compute when to begin the correct motion commands (usually at about 300 msec before the actual grasp). Von Hofsten disputes the use of retinal expansion information because it is clear that people are able to track targets in which there is no such expansion, such as objects that are circling or passing across the field of view. He suggested an alternative schema in which people calculate the distance to a moving object by using the vergence angle to the object. Vision seems to be used predominantly to track the moving object, but the catcher also tracks his hand during reaching to aid his nonvisual proprioceptive senses, that is, to help judge the position of his hand in relation to the environment. Finally, vision must be used during the reaching phase to orient the hand correctly in relation to the object that is being caught.

We also note a relevant fact for human contact and grasping of objects. The central fac-

tor to the final grasp is the time of the onset of hand closure. In early childhood (up to about 5 months), closing the hand is triggered primarily by touch. Children tend to begin grasping only when they are already in contact with the object. By the time a child is 13 months old, however, the hand closes before touch, on average as early as for adults. We take the view below that our robotic system is past early childhood - we will close the hand before actual contact is made.

The initial strategy we have adopted in picking up the object is an open loop strategy, similar in spirit to the pre-programmed motor control schemas described in the psychological literature. Schmidt's schema theory holds that for tasks of short duration, perception is used to find a set of parameters to pass to a motor control program. It is not used during the execution of a task. When grasping a moving object, for example, once vision determined the trajectory of the object, the reach and grasping motor schemas take over with no interference from vision.

In our implementation of this strategy, vision is not used to continually monitor the grasping, but only to provide a final position and velocity from which the arm is directed to very quickly move to the object. This automatic movement is done by establishing coordinate frames of action for each of the components of the system and solving transformation equations.

The transformation equations permit dynamic updating of the arm position by generating the 4×4 transform of the moving object's position from the vision system and sending this information to the arm control algorithm. Because the movement of the hand requires a small amount of time during which the object may have moved, the object's trajectory is predicted ahead during the movement using the $\alpha - \beta - \gamma$ predictor. By keeping the fingers of the hand spread during this maneuver, no actual contact takes place until the gripper reaches the position of the moving object.

6 EXPERIMENTAL RESULTS

We have implemented the system described above in order to demonstrate the capability of the methods. The goal was to track a moving model train, intercept it, stably grasp it and pick it up. The train was moving in an oval trajectory; however, the system had no *a priori* knowledge of this particular trajectory. The setup of our system is presented in figure 1. The velocity of the train was $10 - 20 cm/s$. In this section we present some results obtained by experiments. First, in figure 7 we have the actual measured arclength signal s_1 (black) and the filtered signal s_0 (gray). It is noticeable that s_0 is somewhat *below* the expected value of s_1. The nature of s_1 is quite noisy; however, the analysis described in section 4 was able to accurately extract the correct control signal. The arm control is particularly smooth and jerk free, as well as being accurate enough to intercept and grasp the object between the jaws of the gripper. Figure 8 shows the moving object's trajectory points computed by the vision algorithm (black) and the commanded control signals after filtering (gray). As can be seen, the control system is able to accomplish its task of both smoothing for noise and extracting an accurate position of the moving object.

Because we are using a parallel jaw gripper, the jaws must remain aligned with the tangent to the actual trajectory of the moving object. The system controls the gripper direction (joint 6 on the robot) to be parallel to this tangential direction, allowing grasping to occur at any point in the trajectory.

Figure 9 shows 3 frames taken from a video tape of the system intercepting, grasping and picking up the object (this video tape has been submitted to the video portion of the IEEE Robotics and Automation conference). The system is quite repeatable, and is able to track other arbitrary trajectories in addition to the one shown.

7 FUTURE WORK AND CONCLUSIONS

We have developed a robust system for tracking and grasping moving objects. The system relies on real-time stereo triangulation of optic-flow fields and is able to cope with the inherent noise and inaccuracy of visual sensors by applying parameterized filters that smooth and can

predict ahead the moving object's position. Once this tracking is achieved, a grasping strategy is applied that performs an analog of human arm movement schemas.

Our future work is concerned with implementing other possible grasping strategies. One strategy we are currently exploring is to visually monitor the interception of the hand and object and use this visual information to update the **Drive** transform at video update rates. This approach is computationally more demanding, requiring multiple moving object tracking capability. The initial vision tracking described above is capable of single object tracking only. If we attempt to visually servo the moving robotic arm with the moving object, we have introduced multiple moving objects into the scene.

We have identified 2 possible approaches to tracking these multiple objects visually. The first is to use the PIPE's region of interest operator that can effectively "window" the visual field and compute different motion energies in each window concurrently. Each region can be assigned to a different stage of the PIPE and compute its result independently. This approach assumes that the moving objects can be segmented. This is possible since the motion of the hand in 3-D is known - we have commanded it ourselves. Therefore, since we know the camera parameters and 3-D position of the hand, it will be possible to find the relevant image-space coordinates that correspond to the 3-D position of the hand. Once these are known, we can form a window centered on this position in the PIPE, and concurrently compute motion energy of the moving object and the moving hand in each camera. Each of these motion centroids can then be triangulated to find the effective positions of both the hand and object and compute the new **Drive** transform. Both computations must, however, compete for the hardware histogramming capability needed for centroid computation, and this will effectively reduce the bandwidth of position updating by a factor of 2.

Another approach is to use a coarse-fine hierarchical control system that uses a multi-sensor approach. As we approach the object for grasping, we can shift the visual attention from the static cameras used in 3-D triangulation to a single camera mounted on the wrist of the robotic hand. Once we have determined that the moving object is in the field of view of this camera, we can use its estimates of motion via optic-flow to keep the object to grasped in the center of the wrist camera's field of view. This control information will be used to compute the **Drive** transform to correctly move the hand to intercept the object. We have implemented such a tracking system with a different robotic system [42] and can adapt this method to this particular task.

References

[1] P. J. Burt, J. R. Bergen, R. Hingorani, R. Kolczynski, W. A. Lee, A. Leung, J. Lubin, and H. Shvayster. Object tracking with a moving camera. In *IEEE Workshop on Visual Motion*, pages 2–12, Irvine, CA, March 20–22 1988.

[2] Sang Wook Lee and K. Wohn. Tracking moving objects by a mobile camera. Technical Report MS-CIS-88-97, University of Pennsylvania, Department of Computer and Information Science, Philadelphia, November 1988.

[3] Johan Wiklund and Gosta Granlund. Tracking of multiple moving objects. In V. Cappelini, editor, *Time Varying Image Processing and Moving Object Recognition*, pages 241–249, 1987.

[4] Peter Corke, Richard Paul, and K. Wohn. Video-rate visual servoing for sensory-based robotics. Technical report, GRASP Laboratory, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, 1989.

[5] Richard Goldenberg, Wan Chi Lau, Alfred She, and Alan Waxman. Progress on the prototype pipe. In *IEEE Conference on Robotics and Automation*, Raleigh, N. C., March 31-April 3 1987.

[6] Ren C. Luo, Robert E. Mullen Jr., and Daniel E. Wessell. An adaptive robotic tracking system using optical flow. In *IEEE Conference on Robotics and Automation*, pages 568–573, Philadelphia, 1988.

[7] Gilbert Verghese, Karey Gale Lynch, and Charles R. Dyer. Real-time motion tracking

of three-dimensional objects. In *IEEE International Conference on Robotics and Automation*, Cincinnati, May 13–18 1990.

[8] Reem Bassam Safadi. An adaptive algorithm for robotics and computer vision application. Technical Report MS-CIS-88-05, Department of Computer and Information Science, University of Pennsylvania, January 1988.

[9] B. S. Y. Rao and H. F. Durrant-Whyte. A fully decentralized algorithm for multisensor Kalman filtering. Technical Report OUEL 1787/89, Dept. of Engineering Science, University of Oxford, 1989.

[10] W. Thomas Miller. Real-time application of neural networks for sensor-based control of robots with vision. *IEEE Transactions on Systems, Man and Cybernetics*, 19(4):825–831, July/Aug 1989.

[11] Lee E. Weiss, Arthur Sanderson, and Charles P. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal of Robotics and Automation*, RA-3(5):404–417, October 1987.

[12] Christopher Brown. Gaze controls with interaction delays. *Proc. DARPA Image Understanding Workshop*, pages 200–218, May 23-26 1989.

[13] James J. Clark and Nicola J. Ferrier. Control of visual attention in mobile robots. *IEEE Conference on Robotics and Automation*, pages 826–831, May 15-19, 1989.

[14] P. J. Burt *et al.* Object tracking with a moving camera. In *Proceedings of the IEEE Conference on Robotics and Automation*, 1989.

[15] J. T. Feddema and C. S. George Lee. Adaptive image feature prediction and control for visual tracking with a hand-eye coordinated camera. *IEEE Transaction on Systems, Man and Cybernetics*, 20(5), 1990.

[16] M. Xie. Dynamic vision: Does 3d scene perception necessarily need two cameras or just one? Technical report, Institut National de Recherche en Informatique et en Automatique, 1989.

[17] P. K. Allen, B. Yoshimi, and A. Timcenko. Real-time visual servoing. In *DARPA Image Understanding Workshop*, pages 909–915, 1990.

[18] R. B. Safadi. An adaptive tracking algorithm for robotics and computer vision application. Master's thesis, University of Pennsylvania, 1988.

[19] N. Papanikolopoulos, T. Kanade, and P. Khosla. Vision and control techniques for robotic visual tracking. In *Proceedings of the IEEE Conference on Robotics and Automation*, 1991.

[20] S. Lee and Y. Kay. An accurate estimation of 3d position and orientation of a moving object for robot stereo vision: Kalman filter approach. In *Proceedings of the IEEE Conference on Robotics and Automation*, 1990.

[21] N. A. Andersen, O. Ravn, and A. T. Sorensen. Using vision in real-time control systems. In *American Control Conference*, 1991.

[22] N. Papanikolopoulos, P. K. Khosla, and T. Kanade. Adaptive robotic visual tracking. In *American Control Conference*, 1991.

[23] N. Houshangi. Control of a robotic manipulator to grasp a moving target using vision. In *Proceedings of the IEEE Conference on Robotics and Automation*, 1990.

[24] A. J. Koivo and N. Houshangi. Real-time vision feedback for servoing robotic manipulator with self-tuning controller. *IEEE Transaction on Systems, Man and Cybernetics*, 21(1), 1991.

[25] P. Anandan. Measuring visual motion from image sequences. Technical Report COINS TR-87-21, COINS Dept., University of Massachusetts-Amherst, 1987.

[26] P. J. Burt, C. Yen, and X. Xu. Multiresolution flow-through motion analysis. In *Proceedings of the IEEE CVPR Conference*, pages 246–252, 1983.

[27] G. L. Scott. Four-line method of locally estimating optic flow. *Image and Vision Computing, 5(2)*, 1986.

[28] B. K. P. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1983.

[29] H. H. Nagel. On the estimation of dense displacement vector fields from image sequences. In *Workshop on motion: Representation and Perception*, pages 59–65, Toronto, 1983.

[30] B. F. Buxton and H. Buxton. Computation of optic flow from the motion of edge features in image sequences. *Image and Vision Computing*, 2, 1984.

[31] David Heeger. A model for extraction of image flow. In *First International Conference on Computer Vision*, London, 1987.

[32] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America*, 2(2):284–299, 1985.

[33] E. W. Kent, M. O. Shneier, and R. Lumia. Pipe: Pipelined image processing engine. *Journal of Parallel and Distributed Computing*, (2):50–78, 1985.

[34] L. B. Jackson. *Digital Filters and Signal Processing*. Kluwer Academic Publishers, 1986.

[35] Vincent Hayward and Richard Paul. Robot manipulator control under unix. In *Proc. of the 13th ISIR*, pages 20:32–20:44, Chicago, April 17-21 1983.

[36] Richard Paul. *Robot Manipulators*. MIT Press, Cambridge, MA, 1981.

[37] H. Heuer H. Cruse, J. Dean and R.A. Schmidt. Utilization of sensory information for motor control. In Herbert Heuer and Andries F. Sanders, editors, *Perspectives on Perception and Action*, pages 43–79. Lawrence Erlbaum, 1987.

[38] Claes von Hofsten. Early development of grasping an object in space-time. In Melvyn A. Goodale, editor, *Vision and Action: The Control of Grasping*, pages 65–79. Ablex Publishing Company, 1990.

[39] Claes von Hofsten. Catching. In Herbert Heuer and Andries F. Sanders, editors, *Perspectives on Perception and Action*, pages 33–36. Lawrence Erlbaum Associates, 1987.

[40] Michael Arbib, Thea Iberall, and Damian Lyons. Coordinated control programs for movements of the hand. Technical Report COINS TR 83-25, Dept. of CS University of Massachusetts, August 1983.

[41] D.N. Lee, D.S. Young, P.E. Reddish, S. Lough, and T.M.H Clayton. Visual timing in hitting an accelerating ball. *Quarterly Journal of Experimental Psychology*, 35A:333–346, 1983.

[42] Peter Allen. Real-time motion tracking using spatio-temporal filters. In *Proceedings of DARPA Image Understanding Workshop*, Palo Alto, May 1989.

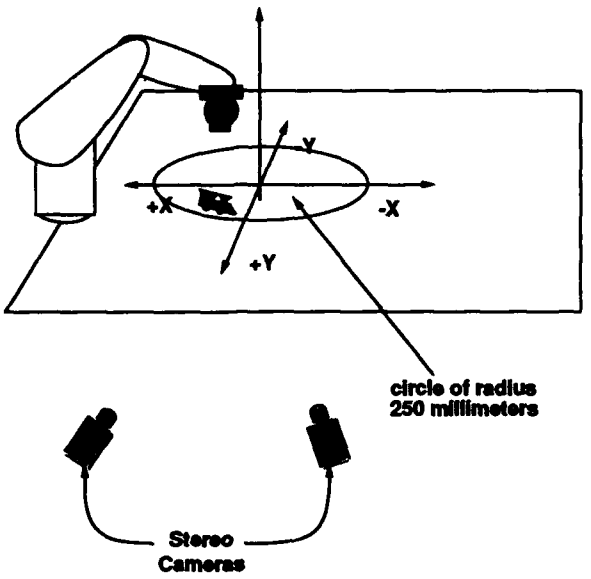


Figure 1: Tracking Grasping System

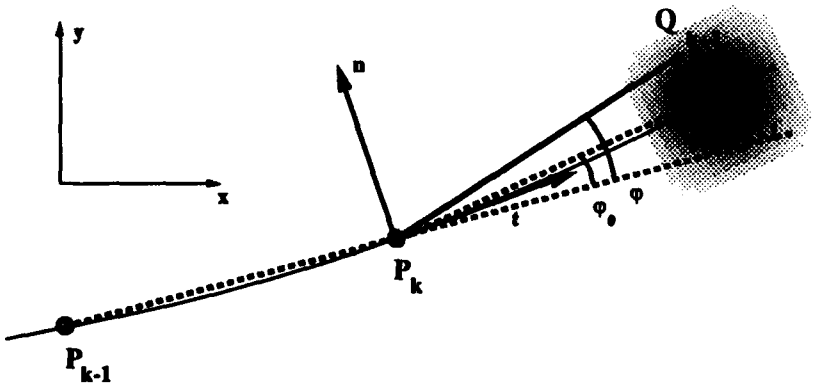


Figure 2: Trajectory: the moving object is in P_{k+1} while the vision computes Q_{k+1}

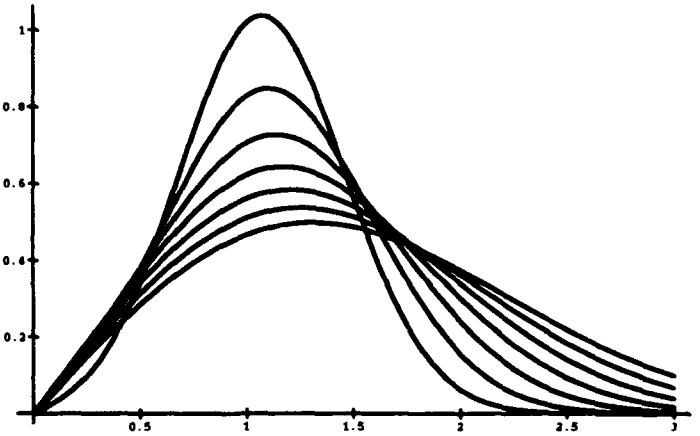


Figure 3: Distribution density $f(s), s_0 = 1, \sigma = 0.4 - 1.0, increment = 0.1$

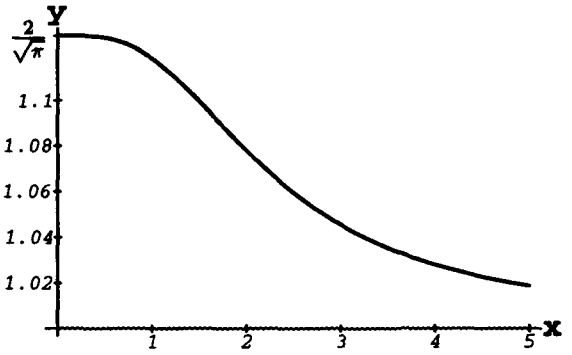


Figure 4: $y = u_1(x)$

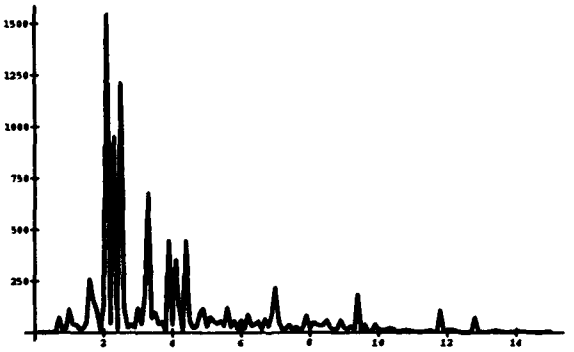


Figure 5: The density of s_1

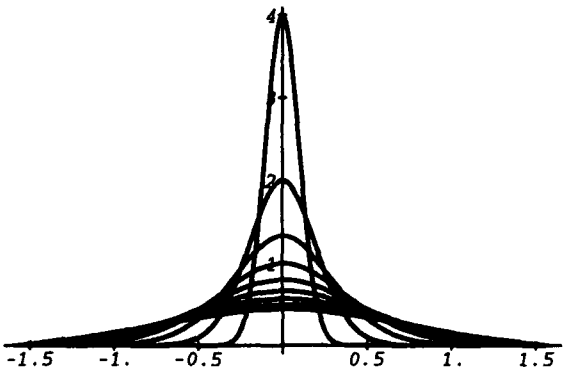


Figure 6: Distribution density $f(\phi)$

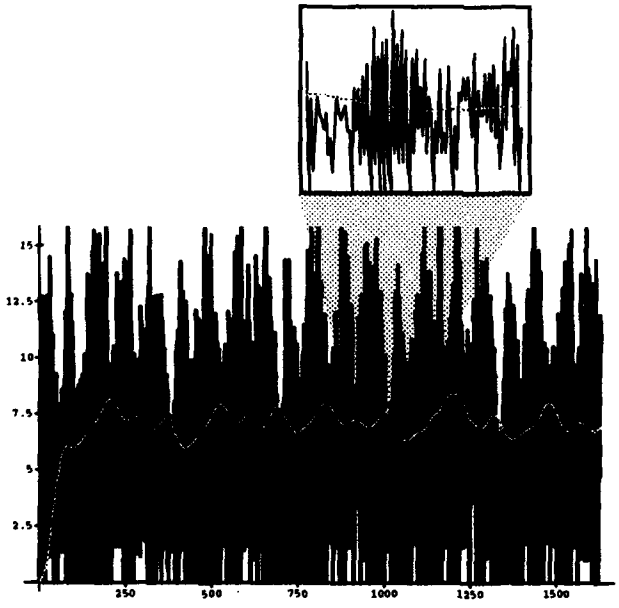


Figure 7: Input signal s_1 (black) and filtered signal s_0 (gray)

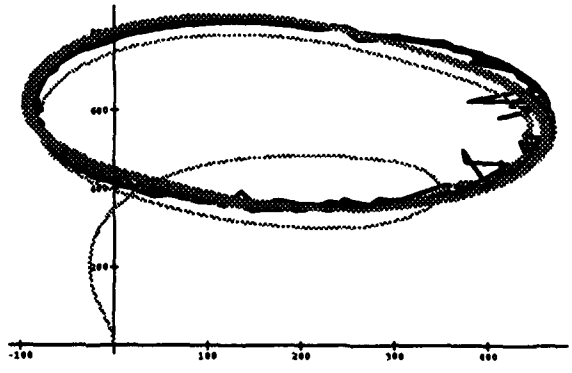


Figure 8: Input trajectory (black) and filtered trajectory (gray)



Figure 9: Intercepting, grasping and picking up the object

Subsection I:

IU Software and Parallel Processing

Overcoming the Barriers to Architecture-Independent Image Processing

Jon A. Webb
School of Computer Science, Carnegie Mellon University
Pittsburgh, PA 15213-3890

By applying simple ideas from parallel programming research, it is possible to overcome the serious barriers in software development to widespread application of parallel computers to image processing. This is being done in the Adapt language [22] project, which exploits these ideas: complete architecture independence; the little language approach; and the simultaneous development of language, parallel implementations, applications, and a significant library of image processing programs, based on the emerging ANSI/ISO standard Programmer's Imaging Kernel System.

1. Introduction

Image processing is ripe for the application of parallel computers. Powerful, reliable parallel computers have been capable of being applied to problems in image processing research for years; and now, image processing computers based on parallelism rather than fixed-function hardware pipelining are becoming available. Indeed parallelism is the only available source of the power that will be required to process images of significant size without inordinate investment in costly, difficult-to-maintain, hardware.

For years people have searched for a solution to the problem of programming these parallel computers. Some have implicitly assumed that a powerful parallel computer would offer significant enough advantage to lure everyone to adopt that machine as a standard, and have introduced architecture-specific languages. But this approach has proved incapable of keeping up with advances in parallel computer architecture; any architecture-specific approach goes out of date rapidly as new parallel computer features are introduced.

Others have introduced architecture independence at various levels—hiding data distribution from the programmer, or managing the distribution of tasks automatically. These approaches do introduce some flexibility, but they do not help the image processing programmer solve problems; they merely reduce the number of new features that must be understood in order to program an architecture, or class of architectures. It is the thesis of this paper that no less than complete hiding of the underlying parallel computer will prove sufficient to successfully replacing serial with parallel computers for development of new image processing algorithms. Anything less creates unacceptable cost to the programmer, and unnecessarily restricts the programs developed. It is possible to achieve this goal through the use of specialized languages for image processing.

This research was supported by the National Science Foundation under Grant MIPS 8920420 and by the Defense Advanced Research Projects Agency and monitored by U. S. Army Training and Doctrine in Fort Huachuca, Arizona under Contract DABT63-91-C-0035.

The Government has certain rights in this material. Any views, opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.They should also not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

Within the design of such languages, the traditional approach has been what we shall call here the *big language* approach. In big languages, objects such as images are manipulated as elementary data objects. This approach has led to a difficult programming style, proliferation of primitive operators, loss of understanding by the programmer of efficiency issues, and difficulty of implementation on MIMD computers.

We reject this approach in favor of what we shall call the *little languages* approach. (The term "little language" comes from an article in which specialized, limited-function languages were first described as a language class [3] .) In this method, a simple language is defined with a straightforward model of mapping a program onto the images to be processed, and then ordinary language constructs are used to describe the operations to be performed on the image elements. This approach leads to a natural programming style (the programmer is essentially writing the inner loop of a computation iterated over the image), introduces no new primitive operators (except those that may be convenient due to the specialized nature of the language), gives the programmer a straightforward understanding of efficiency issues, and can easily be implemented on MIMD computers.

Finally, previous approaches in parallel language design for computer vision have often focussed too strongly on the capabilities of parallel computers while ignoring what is needed for successful computer vision programming, or have concentrated too strongly on the needs of computer vision while ignoring the restrictions that parallel computers impose on the language designer. Only by integrating language design, parallel implementations, applications development, and program library construction can this problem be overcome.

2. Architecture Independence

For the purposes of this paper, we limit the class of machines on which we want to achieve architecture independence to MIMD computers. The reason for this is perfectly straightfoward: advances in processor architecture are being driven at a very high rate for serial processors, because of the large market for successful serial processors. These advances contribute directly to increased performance of MIMD computers, both in hardware and software; for example, future iWarp systems will use standard microprocessor designs together with an additional communications component [6] . The same is not true for SIMD designs; advances there are driven only by the much smaller investment in SIMD computers.

Now we can consider various levels of architecture independence:

1. **Processor Architecture.** This is the degree of architecture independence provided by conventional serial processor languages. For example, the number of registers, the presence or absence of a math coprocessor, etc. are hidden.

2. **Array Size.** The size of the processor array is hidden: the array topology must be the same, as well as the intercommunication scheme. This kind of independence is extremely useful—it allows the development of code on a small array of nodes, and production runs on large arrays, as well as upgrades of processor arrays to larger systems without changing software—but it does not allow the movement of code from one architecture to another.

3. **Array Topology.** This level supports changes in the shape of the underlying processor array as well as its size. Processes can be placed on whatever processors are convenient, and the programming system takes care of managing communication using whatever array topology is available.

4. **Interprocessor Communication.** This level completely hides interprocessor

communication. For example, Linda [9] manages interprocessor communication by supporting a "tuple" model in which processes place tuples in a common database and read them from there. The Uniform System [15] also provides this level of architecture independence.

5. **Process Management.** This is the highest level of architecture independence. The existence of multiple processes is hidden from the programmer, as well as all communication. The program uses a model that does not make any mention of multiple processes, and the compiler takes care of splitting the work into multiple processes and allocation of data. This potentially allows MIMD. serial, and even SIMD architectures to be programmed efficiently with the same program.

It should be clear that the image processing programmer wants the highest degree of architecture independence described above; anything less leads to a reduction of the effort applied to solving image processing problems, and an increase in the effort in solving the problems of parallel processing. Solving these problems may be interesting in their own right, but to the image processing programmer they are of no more concern than figuring out how to manipulate images on a machine whose physical memory is too small for an image to fit. When a general solution is found for such a problem, such as virtual memory, this leads to increased productivity and a decrease in machine-specific solutions.

Now, in the case of a general programming language such as C, it may well prove impossible to provide the programmer with a high degree of architecture independence. But with specialized languages, particularly collection-oriented languages that exploit data parallelism, complete architecture independence is not only achievable, but not difficult to achieve.

Experience with the Apply language shows just how easy it can be to achieve this degree of architecture independence. Apply is specialized for local image processing operations, such as edge detection, convolution, smoothing, point operations, and so on. Apply provides the highest degree of architecture independence above. Now, over a period of a few years Apply compilers were developed for all of these machines:

- Warp (10 processor one-dimensional systolic array) [17] , iWarp (4-1024 processor two-dimensional systolic array) [2] , FT Warp (two-dimensional systolic array with fault-tolerance) [11] .
- Sun and other UNIX architectures (serial architecture) [17] .
- Carnegie Mellon SLAP (a SIMD machine one processor per column in an image—e.g., 512 processors for a 512×512 image.) [8]
- University of Massachusetts Image Understanding Architecture (one processor per pixel in the largest implementation) [14] .
- Meiko Computing Surface (transputer-based, reconfigurable) [18] .
- Hughes HBA, a video bus-based machine [16, 17] .1

None of these efforts involved more than two people actively programming the compiler, or more than a few months of programming time to get the compiler working efficiently. The implementations span the gamut of computers being used for parallel image processing today, including even SIMD machines. Yes programs can be run efficiently on any of these machines, simply by recompiling the Apply code.

The reason these implementation efforts have been so successful with so little effort is

the use of the *little languages* model, as will now be explained.

3. Little Languages

There are two kinds of specialized languages for image processing: those that manipulate the images as a whole, which we call *big* languages, and those that map over the image a serial program that manipulates images elements, which we call *little* languages. The term "big" comes from the large data structures manipulated by such languages; "little" comes from the contrast to big, and an article [3] in which such specialized languages were first described as a distinct language class.

A trivial example will illustrate the distinction between the two approaches. Suppose a 2×2 local averaging operation is to be programmed. In a big language, the program would perform these steps: shift the image one pixel to the right; add the shifted image to the original; shift this result one pixel down; add this to the previous result; divide by four. In a little language, such as Apply, the programmer would state in the procedure header that a 2×2 window was to be taken on the input image. The program would assign to the output image the sum of the four pixels in this window, divided by four.

Now, big languages are the more popular of the two approaches. Many big languages for image processing have been constructed, deriving their design from image algebra [13] , for example. But in terms of taking advantage of the capabilities of MIMD parallel computers, these languages suffer from several defects:

- The programmer is forced to write in an unfamiliar and awkward programming style. This is best illustrated by the difficulty of writing highly conditional local image processing operators in such languages: for example, the median filter. Median filter requires calculating the median locally at every pixel in the image, an operation that involves comparison of image values with neighbors and the use of some local storage. In a big language, the images must be shifted and manipulated as a whole, making it difficult to write code that must take the separate paths through conditional expressions that are necessary to calculate the median.

- Big languages include a lot of primitive operators, which must be learned. This imposes a heavy start-up cost for the programmer. Applications must be completely reprogrammed as well. Little language programs are written using familiar operators, so that learning is easy, and it is even possible to reuse code from serial implementations with only minor changes.

- Big languages can easily be compiled for SIMD machines, because the tight coupling between processors makes it possible to issue each primitive operator as an instruction sequence to the entire array. This is not so on MIMD machines; since the processors there are more loosely coupled, a similar process would be woefully inefficient. Instead, a process known as clustering must be employed [4] . Clustering combines a sequence of primitive operators into a single loop. This process requires careful compile time analysis, including estimates of the sizes of the objects to which the primitives are applied. Little languages can be compiled efficiently through the use of simple template-based code generation techniques.

- Clustering requires a lot of information about the primitives being clustered together, making it difficult to deal with procedure call in big languages, especially if the procedure being called has been compiled separately. Procedure call is trivial

```
procedure egsbl(imagein   : in image array (-1..1, -1..1) of byte border 0,
                type      : integer,
                imageout  : out image byte)
is next
 horz, vert : integer;
begin
 horz := imagein(-1,-1) + 2*imagein(-1, 0) + imagein(-1, 1) -
         imagein( 1,-1) - 2*imagein( 1, 0) - imagein( 1, 1);
 vert := imagein(-1,-1) + 2*imagein( 0,-1) + imagein( 1,-1) -
         imagein(-1, 1) - 2*imagein( 0, 1) - imagein( 1, 1);
 if type = 1  then imageout := sqrt(horz*horz + vert*vert);
              else imageout := abs(horz) + abs(vert);
 end if;
end next;
end egsbl;
```

Figure 1. Adapt Sobel Operator

```
procedure histogram( im: in image byte, hist: out array(0..255) of float)
is
  count: array(0..255) of integer;
first begin
  for i in 0..255 loop count(i) := 0; end loop;
end first;
next begin
  count(im) := count(im) + 1;
end next;
combine begin
  for i in 0..255 loop count(i) := count(i)+_count(i); end loop;
end combine;
last
  pixels: integer;
begin
  pixels := 0;
  for i in 0..255 loop pixels := pixels + count(i); end loop;
  for i in 0..255 loop hist(i) := float(count(i))/pixels; end loop;
end last;
end histogram;
```

Figure 2. Adapt Histogram

to deal with in little languages, even to separately compiled procedures.

- Little languages can support programming models especially designed for the applications domain, while big languages tend to support only a few uniform models. For example, Adapt's fundamental programming model is that an operation is applied once per pixel in parallel. But, within a row, previous results can be reused, making it easy to program operators that take advantage of locality.

- The optimizations the big language compiler must go through in generating efficient MIMD code makes the resulting program very different from the original big language program. As a result, the programmer may not have a clear model of the efficiency issues involved in the program; subtle changes in the big language code can make a big difference in performance on the MIMD machine. In contrast, template-based code generation in little languages gives a program that looks very much like the original, making the efficiency issues much easier for the programmer to deal with.

- For the same reason, designing a debugger for big language programs on MIMD computers is quite hard. A little language debugger would only have to undo some simple source transformation present the programmer with a model of the code being

executed that corresponds exactly to the little language program originally written.

Let us examine the process of compilation of Adapt programs to understand how this can be done. An Adapt procedure consists of four sections:

First An initialization function, which may be run at the beginning of any row of the image.

Next This section is applied in raster order across the image (wrapped around at the borders of the image). Each execution of *Next* is guaranteed to be preceded either by another execution of *Next* (for the previous pixel), or by *First* at the beginning of a row. Pixels in the *Next* section are referenced relative to the current pixel position, as illustrated by the Sobel operator program in Figure 1.

Combine A merging function, which combines the outputs of any two image regions to produce an output for the concatenation of the two regions. To make programming easier, *Combine* will be applied only to adjacent groups of consecutive rows of the image. *Combine* can reference any variable used elsewhere in the program, which retain their values. Its output is also expressed in terms of these variables. Within the *Combine* section, variables preceded by an underscore refer to the values of variables in the lower image region, and variables not preceded by an underscore refer to values from the upper image region. The *Combine* section result is to assign to variables not preceded by an underscore the correct values for the merged region.

Last A termination function, which is applied once after the output of the entire image is computed.

An Adapt program for image histogram is shown in Figure 2. The *First* section (lines 17 through 19) zeroes the histogram, the *Next* section (lines 20 through 22) increments the histogram element for the current pixel, the *Combine* section (lines 23 through 25) adds two histograms together, and the *Last* section (lines 26 through 32) divides the histogram by the total pixel count to create a pixel frequency array.

In order to compile an Adapt program for a MIMD computer, the compiler must go through these steps:

1. *Distribute images.* Images are divided among processors in some regular pattern, usually as swaths of rows. The images must be divided and distributed to processors. At this level of vision, processing is unlikely to vary significantly across the image, so it is usually sufficient to simply give each processor an equal-sized swath. But load balancing techniques can be used; for example, an initial set of swaths can be dealt out to processors, and then new swaths can be provided as processors finish their work. It is also possible to distribute a few rows of the images to processors at a time, which will allow processors to overlap input, output, and computation on machines with that capability, for example systolic arrays. Note that it is also possible to leave the images distributed over the processor array, so that all that is necessary for a processor to get its input images is for that processor to exchange a few rows with its neighbors, to cover the window of the image being processed.

2. *Distribute other inputs.* These can be distributed to processors via a simple broadcast mechanism.

3. *Map relative addressing onto image buffers.* The *First* and *Next* sections use addressing relative to the "current" pixel. The compiler must replace these relative addresses with whatever array reference is needed to access the proper pixels in the image buffers. This is essentially a macro-expansion.

4. *Generate First section.* The *First* section can be executed immediately once the inputs for a processor are available.

5. *Embed Next section.* The *Next* section must be embedded within appropriate `for` loops that will cause it to be iterated over the section of the image allocated to a processor.

6. *Collect images.* The output images are distributed in the same way as the input images, and must be collected similarly.

7. *Extract Combine variables.* The compiler must scan the *Combine* section to determine which variables must actually be passed among processors. In theory, dataflow analysis would be required to determine which variables are live within the *Combine* section, but in practice this yields little improvement in performance, since usually the only variables that are eliminated by this technique are scalars, and scalars take up little space in a message in any case. The extracted *Combine* variables must be packed within a message, and references to them must replaced with references to the appropriate message component, again via macro-expansion.

8. *Generate Combining tree.* The pattern by which a processor combines its results—serially, in a two-dimensional pattern, or in a binary tree—depends on the message-passing capabilities of the underlying processor. This pattern is determined for the particular machine by the Adapt compiler developer, and the compiler simply generates the appropriate code to pass the necessary *Combine* variables among processors, then generates the *Combine* section applied to the two sets of *Combine* variables. At the top level of this tree, the *Combine* variables are passed back to the calling processor.

9. *Generate Last section.* The code on the calling processor executes the *Last* section, applied to the *Combine* variables returned by the combining tree.

Thus, the process of compiling an Adapt program onto a target machine comes down to choosing an appropriate I/O model, and making the appropriate macro-substitutions in Adapt programs to translate the code into the target language with image and combine variables appropriately changed. As a result, given an Adapt parser, generating an Adapt compiler is usually a matter of a few weeks of programming.

The resulting programs are efficient, because the I/O model is designed for the particular target machine, and because the compiler makes so few modifications to the user's program. Experience with serial languages has given the user a good idea of what is necessary to design code efficiently, and this experience is directly useful in Adapt.

Nothing in this simple compilation step requires any special processing for procedure call within Adapt programs; even procedures of foreign languages can be called, so long as their argument passing conventions are compatible with Adapt. For example, the iWarp Adapt implementation uses the standard Unix math library without change, simply by providing an Adapt procedure header for each Unix math routine in an Adapt header file.

4. Integrated Language Development

The Warp project showed the value of integrated architecture, applications, and software development [7] . Integrated development helps ensure that the system developed solves problems of importance to the application, exposes new and interesting problems to be solved, and helps ensure that the system developed will actually be used in the targeted application.

We have followed this model in the development of the Adapt language, by simultaneously addressing the issues of algorithm, architecture, applications, and libraries. From the earliest days of design of the language, we have studied important algorithms such as connected components, Hough transform, image warping, and run-length encodin [21, 20, 10] g to make sure that the Adapt model can be used to implement such algorithms. In cases where important image processing algorithms cannot be implemented in the Adapt model, such as error-diffusion halftoning, we have devised new algorithms [19] that can be implemented in Adapt.

In addressing architectural considerations, we have implemented the language on a variety of computers. Adapt implementations exist for Unix/C architectures, the Carnegie Mellon Nectar compute [1] r, the Carnegie Mellon Warp computer, and the Intel-Carnegie Mellon iWarp computer. Each of these designs has exploited features specialized to the particular architecture. For example, the Nectar implementation does automatic load balancing by allocating a small slice to each processor, then sending more as processors complete. Lightly loaded processors will end up doing more work in this model, and heavily loaded processors will do only the initial small slice. On Warp, there are two separate implementations: one which partitioned the image by rows, and another which partitioned the image by columns. We found that the row-partitioned method was much faster, because of reduced coupling between processors and reduced I/O when sending intermediate results. The iWarp implementation exploits iWarp's logical pathways when constructing the combining network; even though iWarp is physically a two-dimensional mesh, intermediate results are combined in a binary tree.

We also have an implementation on the Intel Touchstone computer underway. This will take advantage of the Mach operating systems on the Touchstone, which is being implemented at Carnegie Mellon. Images will be stored distributed among processors, and processors will get the data they need for computing their local result by automatic message passing generated through access to memory locations actually stored in other processors.

The Sun implementation of Adapt has proved particularly successful for code development. The Adapt Sun implementation is not an Adapt simulation; Adapt produces good code, comparable to the best hand-written code for serial image processing. Adapt programs on the Sun can be debugged completely, then only a simple recompilation step is necessary to port the code to iWarp. This greatly speeds up code development time, as well as reducing the load on the iWarp for debugging. Moreover, there is a "parallel simulation" mode in the Sun Adapt compiler that allows the user to run the program as if it was being run on a parallel processor (at a small loss of efficiency). This allows the programmer to uncover bugs in the *Combine* section of the Adapt code on the Sun.

Applications have also been studied in the Adapt language from the beginning. The parallel halftoning algorithm mentioned earlier yields high-quality halftoned images (higher quality than error-diffusion methods), and this has been exploited in a program for halftoning portrait images for display on monochrome big-mapped displays. We have done work on image spectrogram computation and feature extraction modules for an object recognition system. Currently in progress are implementations of the second DARPA image understanding benchmark, research on magnetic resonance image reconstruction, development

of a stereo vision algorithm, and programming the Joint Photographic Experts Group (JPEG) image compression algorithm in Adapt.

The Adapt library development has proceeded in two stages. First, the existing WEB library (a library of local image processing operations for the Apply language) has been translated to Adapt and has been verified to work with Adapt. Second, a more ambitious library development effort is underway, based on the ANSI/ISO Programmer's Imaging Kernel System (PIKS).

The PIKS library design is a result of meetings of the ANSI X3H3.8 Imaging Applications Program Interface task group, a group of engineers and researchers largely from industry, with representation from Sun, Mitre, Wang, Kodak, Datacube, and so on. The PIKS library has been proposed for ISO status, and current plans are for the library to be in nearly final form at the beginning of 1992.

The library includes about 200 different operators, covering these applications areas [5] :

- primitive image manipulation
- image enhancement
- image restoration
- image analysis
- image classification (basic)
- image visualization (basic)
- standard color models
- image transport
- image compression and decompression

The library will be implemented in two stages. Initially, we will concentrate on the PIKS Core routines, which are the routines that cover most of the functionality in PIKS. These are the following [12] :

- **Analysis operators**: accumulator, amplitude projection, extrema, distance, feature list, line profile, and window statistics.
- **Color processing operators**: additive color conversion, color lookup table, subtractive color conversion, trichromatic conversion.
- **Complex image operators:** complex conjugate, complex magnitude, complex-to-polar, and polar-to-complex.
- **Ensemble operators:** dyadic.
- **Filtering operators:** convolve, two-dimensional, linear filtering, max, min, and median filtering.
- **Geometric operators:** flip, spin, transpose, resize, rotate, subsample, translate, warp (lookup table and polynomial).
- **Histogram operators:** histogram, one-dimensional.
- **Morphological operators:** erosion and dilation (conditional and unconditional), fill contour, hit or miss transformation, and morphic.
- **Point operators:** bit shift, lookup table, monadic, threshold, unary, and window-level.
- **Presentation operators:** dither.
- **Unitary transform operators:** Fourier transform.
- **3D specific operators:** 3D slice.

Following the implementation of the PIKS core, we will move on to implement the complete PIKS library.

The library implementation effort serves three goals in our language development effort:

1. **Completeness.** By implementing a large library of routines that covers so much of image processing, we will ensure that the Adapt language can actually be used for development of practical image processing systems.

2. **Standardization.** The PIKS library promises to be developed and used by a wide variety of image processing software and hardware development companies. By implementing it, we ensure interchangeability of code with applications developed on other platforms.

3. **Software base.** The PIKS library will serve as a basic set of routines for the Adapt programmer to use. Code from the library can be used as a model for development of new routines of similar function.

5. Summary

Reliable parallel hardware is available for image processing, but the full potential of this hardware has not yet been realized. Software has been a bottleneck. We trace this problem to three sources:

1. Lack of complete architecture independence. Anything less than complete hiding of the underlying hardware leads to the introduction of programming issues that are not relevant to the concerns of the image processing programmer, and which make the program limited to run on a particular architecture.

2. Big languages. Big languages, which manipulate images as a whole, are appropriate for SIMD computers. But they impose an unnatural style on the image processing program, especially when dealing with conditional neighborhood operations such as median filter. And it is difficult to compile these languages for MIMD computers, since multiple primitive operations must be clustered together for efficiency.

3. Non-integrated language development. Language development should benefit from the beginning from algorithm, architecture, application, and library experience. Without this some important algorithms may not be implementable in the language, or may be inefficient on an important architecture class, or miss important functionality for the development of working systems.

The Adapt language and its associated PIKS library seeks to overcome these problems. It offers complete architectural independence. It is a little language, which provides a natural model to the programmer that is easy to compile for MIMD machines. And the language development effort is integrated, combining algorithm, architecture, application, and library development.

Bibliography

[1] Arnould, E. A., Bitz, F. J., Cooper, E. C., Kung, H. T., Sansom, R. and Steenkiste, P. A. The Design of Nectar: A Network Backplane for Heterogeneous Multicomputers. *Proceedings of Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS III)*. ACM, April, 1989.

[2] Baxter, B., Cox, G., Gross, T., Kung, H. T., O'Hallaron, D., Peterson, C., Webb, J., and Wiley, P. Building Blocks for a New Generation of Application-Specific Computing Systems. *International Conference on Application Specific Array Processors*. Princeton, NJ, September, 1990.

[3] Bentley, J. Little Languages. *Communications of the ACM*. 29(8):711-721, August, 1986.

[4] Chatterjee, S., Blelloch, G., and Fisher, A. Size and Access Inference for Data-Parallel Programs. *Proceedings of the ACM SIGPLAN '91 Conference on Programming Language Design and Implementation*, pages 130-144. Toronto, Canada, June, 1991.

[5] Clark, A. Image Processing and Interchange (IPI), Part 1: Overview, Architecture, Profiles, and Conformance; Working Draft. ISO/IEC JTC1/SC24/WG1 Imaging Rapporteur Group Document IM 101. September, 1991.

[6] Cox, G. Directions for iWarp Technology. iWarp Forum, Hyatt Regency, Crystal City, Virginia, September 11, 1991.

[7] Crisman, J. D. and Webb, J. A. Warp on Navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 13(5):451-465, May, 1991.

[8] Fisher, A., Highnam, P. T., and Leon, J. Design and Performance of a SIMD Optimizing Compiler. *Third Symposium on the Frontiers of Massively Parallel Computing*. 1990.

[9] Gelertner, D. Generative Communication in Linda. *ACM Transations on Programming Languages and Systems*. 7(1):80-112, January, 1985.

[10]Kistler, J., and Webb, J. A. Connected Components with Split and Merge. *International Parallel Processing Symposium*. Anaheim, Ca, 1991.

[11]Menzilcioglu, O. *Using Powerful Processors in a Configurable Systolic Array Architecture*. PhD thesis, Carnegie Mellon University, October, 1988.

[12]Pratt, W. Image Processing and Interchange (IPI), Part 2: Programmer's Imaging Kernel System (PIKS); Working Draft. ISO/IEC JTC1/SC24/WG1 Imaging Rapporteur Group Document IM 102. September, 1991.

[13]Ridder, G. X., and Gader, P. D. Image algebra techniques for parallel image processing. *Journal of Parallel and Distributed Computing*. 47-44, 1987.

[14]Scudder, M. *An Apply Compiler for the CAAPP*. Technical Report, University of Massachusetts at Amherst, 1990.

[15]Thomas, R. H., and Crowther, W. The Uniform System: An approach to runtime support for large scale shared memory parallel processors. *International Conference on Parallel Processing*, pages 245-254. University Park, PA, 1988.

[16]Wallace, R. S. and Howard, M. D. HBA Vision Architecture: Built and Benchmarked. *Computer Architecures for Pattern Analysis and Machine Intelligence*. IEEE Computer Society, Seattle, Washington, December, 1987.

[17]Wallace, R. S., Webb, J. A. and Wu, I-C. Architecture Independent Image Processing: Performance of Apply on Diverse Architectures. , pages 265-276. 1989.

[18]Wang, H., Dew, P. M., and Webb, J. A. Implementation of Apply on a Transputer Array. *Concurrency: Practice and Experience*. 3(1):43-54, 1991.

[19]Webb, J. A. Accurate Parallel Digital Halftoning. *Society for Information Display.* Anaheim, CA, 1991.

[20]Webb, J. A. Architecture-Independent Global Image Processing. *Tenth International Conference on Pattern Recognition*, pages 623-628. International Assocation for Pattern Recognition, Atlantic City, NJ, June, 1990.

[21]Webb, J. A., and MacPherson, M. The Second DARPA Image Understanding Benchmark on WARP and Extending Apply to Include Global Operations. *Proceedings, Image Understanding Workshop*, pages 597-616. Palo Alto, CA, May, 1989.

[22]Webb, J. A. Apply and Adapt: Towards Architecture Independent Image Processing. *IEEE Computer.* To appear, 1992.

Parallel Linear Feature Extraction

Craig Reinhart*and Ramakant Nevatia†
Institute for Robotics and Intelligent Systems
Departments of Electrical Engineering and Computer Science
University of Southern California
Los Angeles, California 90089-0273

Abstract

In this paper we describe a parallel implementation of the linear feature extraction process. Linear feature extraction is a fundamental step in many computer vision systems. It is the step that "bridges the gap" between iconic and symbolic processing. The task is inherently heterogeneous in that it comprises several processing steps with varying degrees of algorithm complexity and multiple data structures. Such tasks are common among computer vision systems but, unfortunately do not lend themselves to straight forward parallel implementations. We show that implementation designs must not only address the algorithmic requirements, but also the algorithm-to-algorithm interfaces.

1 Introduction

Linear feature extraction (extraction of curves and possible approximation with linear line segments) is a basic operation in most computer vision systems. This is normally thought of as a "low-level" operation, easily implemented in parallel on virtually any parallel machine. However, this is not entirely true. Linear feature extraction is more appropriately viewed as a bridge between low and mid-level processing. The input to this process is an image, but the output is a symbolic structure. We found the paucity of parallel implementation studies that actually aim at outputing the symbolic structure to be surprising. This paper is an effort to correct this situation.

The process of linear feature extraction requires several steps. Typical steps are edge detection, edge linking to yield contours, and approximation of the contour by straight lines (or splines). For some of these steps, the processing is *iconic*, i.e. the input and output are both in the image form. Further, the processing is rather local. In these steps, parallelization is easily achieved on almost any parallel machine. However, at some stage, the information is converted to a *list* form, such as the list of edges forming the contours, or the list of straight lines that approximate the contours. Here, we have a basic transformation of data structures from the iconic to the symbolic. It is this transformation that is the source of difficulty in parallelizing linear feature extraction and is explored in detail in this paper.

2 Background

We consider a specific algorithm in our study [Nevatia and Babu, 1980] however, our method should apply to any other algorithm that follows similar steps. That is, various algorithms have been developed to perform the constituent processing steps. For example, Canny [Canny, 1986] and Marr/Hildreth [Marr and Hildreth, 1980] describe alternative edge detection schemes and Canny [Canny, 1983] describes and alternative approach to contour extraction but these approaches do not alter the overall structure of the linear feature extraction task. Furthermore, the basic operations employed by the alternative approaches are similar, *e.g.* convolutional processing for edge detection. Therefore, we claim that our results are generalizable.

Various researchers have presented parallel implementations of stand-alone edge detection algorithms [Little *et al.*, 1987] [Lee and Aggarwal, 1987] [Weems *et al.*, 1991] while others have concerned themselves with "line finders" that identify the presence of a line, via a Hough transform, for instance, but do not actually extract the line [Weems *et al.*, 1991] [Guerra and Hambrusch, 1989]. Few have investigated parallel implementation of the entire linear feature extraction process. In [Shu *et al.*, 1990] the authors present a fine grain parallel implementation of the entire process on the Image Understanding Architecture [Weems and Levitan, 1987]. The implementation utilizes custom designed hardware and therefore, is unintuitive and somewhat difficult to understand. Thus, it will most likely be difficult to implement (code and debug) and maintain (modify).

In [Vaillant *et al.*, 1989] the authors present a coarse grain parallel implementation using a small number of powerful MIMD processing elements. The algorithm is

*Supported by the Hughes Aircraft Company Fellowship Program

†This research was conducted at the University of Southern California and supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-90-C-0078. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

of a general nature and does not include extensive details specific to the architecture except for interprocess communication. That is, if agrees with one's intuition. A drawback of the implementation is that they do not achieve the full potential algorithm speedup available in the low-level operations which are inherently fine grain and map well to a massively parallel SIMD machine.

Our approach to developing a parallel implementation of the linear feature extraction process is to utilize a heterogeneous pyramid architecture that comprises both a SIMD (fine grain) mesh connected section and a MIMD (coarse grain) ring connected section. This is similar to the scheme incorporated in the Image Understanding Architecture with two exceptions: 1) we do not use the custom gated-connection network; and 2) we use only a ring topology in the MIMD level.The heterogeneous architecture is depicted in figure 1. It is, in a sense, a subset of the IUA.

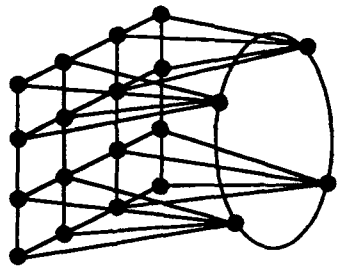


Figure 1: Heterogeneous architecture for the linear feature extraction algorithm.

We justify the use of this architecture by analyzing the system in terms of the methodology described in [Reinhart and Nevatia, 1990] which allows a designer to match parallel architectures to the inherent parallelism of the algorithm.

In the following sections we describe the linear feature extraction process. We briefly describe the algorithm driven methodology used to analyze the process and justify the heterogeneous architecture. We analyze each of the individual algorithms that constitute the process and discuss how those individual results fit into the proposed parallel architecture. Finally, we present results of simulation of our implementation and summarize the work.

3 Process Description

The objective of the linear feature extraction process is to extract linear segments from the input image. Input to the process is the 2-D image array of pixels and output is a list of linear segments with attributes of end-point locations, length, orientation, and contrast. The approach employed is one of multiple (heterogeneous) steps including edge detection, edge thinning, edge linking, contour extraction, and piecewise linear approximation. Details of a typical entire process can be found in [Nevatia and Babu, 1980]. Briefly stated, the following steps are performed:

- *Edge detection* – convolve the input image with a kernel or set of kernel masks.
- *Edge thinning* – compare each edge to its neighbors (in the directions orthogonal to the edge's orientation) and retain the edge if its magnitude is greater than the magnitudes of its neighbors and greater than a fixed threshold.
- *Edge linking* – compare each edge to its neighbors (in the direction of the edge's orientation) and form a link to the neighbors if they are of similar orientation.
- *Contour extraction* – extract (from the 2-D image array) the edge locations ((x, y) coordinates) and save them in an ordered (contour) list.
- *Linear approximation* – join the end points of the contour list with a single line approximation, mark the point of maximum error for this approximation (creating two approximating line segments), iterate the process on the two new segments until the maximum error is within an acceptable bound.

The processing steps and resultant data representations are depicted in figure 2.

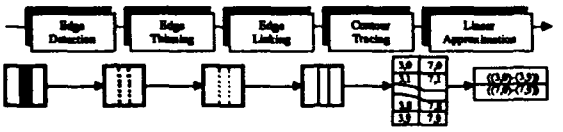


Figure 2: Processing steps and data representations of the linear feature extraction process.

Of primary interest is the contour extraction step. It is this step that actually bridges the gap between the iconic (pixel) representation and the symbolic (linear segment) representation. Prior to this step, data is represented as numbers in a 2-D array. Following this step, data is represented as (x, y) coordinates in a linked-list. We show, in the following sections, that this is also the step that poses the greatest challenge in developing a parallel implementation.

4 The Parallel Implementation

4.1 Overview

The methodology we employ for justifying our choice of a heterogeneous pyramid architecture for the linear feature extraction process is algorithm driven [Reinhart and Nevatia, 1990]. We analyze each algorithm, identifying its inherent parallel characteristics, then show how those characteristics map onto the pyramid. Our analysis proceeds in four steps:

- *Control Structure Analysis* – identify the independent processes that constitute the algorithm.
- *Data Structure Analysis* – determine the data requirements for each of the independent processes.
- *Communication Analysis* – determine the communication requirements between the independent processes.

Algorithm	Complexity	Comments
Edge Detection	$O(i^2)$	$i : i \times i$ pix img
Edge Thinning	$O(i^2)$	$i : i \times i$ pix img
Edge Linking	$O(i^2)$	$i : i \times i$ pix img
Contour Extraction	$O(i^2 l)$	$i : i \times i$ pix img l : avg contour len
Linear Aproximation	$O(nl)$	l : avg contour len n : num of contours

Table 1: Time complexities of the steps that constitute the linear feature extraction process.

- *Architecture Specification* – describe a parallel processor architecture that is well suited to the processing requirements determined via the previous steps.

In using such an approach our goal is to show, analytically, that the implementation will achieve significant algorithm speedup and processor efficiency through algorithms and software that reflect the structure of the "typical" solution to the problem of linear feature extraction. We want to avoid the use of custom hardware and unintuitive algorithm designs as these tend to be costly over the life time of the system.

4.2 Control Structure Analysis

In the control structure analysis step our objective is to identify the primary sources of time complexity within the algorithm and determine which processes that constitute those sources can be performed in parallel.

Table 1 shows the time complexities of each of the constituent algorithms for the linear feature extraction process. The time complexities are predicated on the following concepts. The edge detection, edge thinning, and edge linking steps require every pixel in the image to be processed in a uniform manner. Edge detection requires a convolution with a fixed size kernel, edge thinning and linking require comparisons of an edge with each of its neighbors. The contour extraction step requires a scan of the image plane to detect contour start points followed by a traversal of each contour from its start to its end point. The linear approximation step requires traversing the length of each contour detecting points of inflection.

The primary control structures for each processing step are presented in the following pseudo code.

```
(edge detection)
for each pixel
  convolve with kernel
(edge thinning)
for each edge
  compare neighbor edges and retain
    if it is of greater magnitude
(edge linking)
for each edge
  link to neighbor edges of similar
    orientation to form contours
(contour extraction)
for each contour
  extract constituent edge coord
    to form contour lists
(linear approximation)
for each contour list
  fit piece-wise linear segments
```

In the edge detection, edge thinning, and edge linking steps each image pixel location can be processed independently and synchronously, that is, the same processing steps are applied to each location. The nature of the parallelism is fine grain. In the contour extraction and linear approximation steps, each contour can be processed independently but processing is data dependent and therefore, processed asynchronously. Here, the parallelism is coarse grain.

4.3 Data Structure Analysis

In the data structure analysis step our objective is to identify the input data requirements of each of the independent processes identified by the control structure analysis and to define an appropriate partitioning of the primary data structures.

The input data structure to the linear feature extraction algorithm is the 2-D image array. This is the primary data structure of the edge detection, edge thinning, and edge linking steps. If each pixel location is to be processed independently, then each process requires access to the values of its neighboring locations.

The input data structure to the contour extraction step is the 2-D array containing the linked edges. The resultant data structure is multiple linked-lists of (x, y) coordinates, one linked-list for each contour. If each contour is to be processed (extracted) independently, then each process requires access to the entire 2-D array since a single contour may traverse any part of the array.

The linear approximation step accepts, as input, the linked-lists of (x, y) coordinates, the contours. To process each contour independently a process only needs access to its own contour. Output from the linear approximation step is a list of linear segments represented by their end points, orientation, contrast, and length.

Figure 3 depicts the two primary data structures used by the linear feature extraction process.

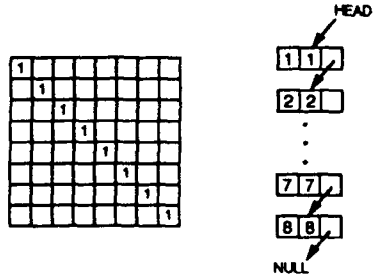


Figure 3: Iconic and symbolic data representations (structures) used in the linear feature extraction process.

4.4 Communication Analysis

In the communication analysis step our objective is to identify the communication requirements among the in-

dependent processes determined by the control structure analysis. We attempt to generalize the requirements into classes such as point-to-point send/receive, broadcast, reduction, shuffle, local, global...

Communication among independent processes of the edge detection, edge thinning, and edge linking steps is restricted to local neighborhoods. Each process must obtain the pixel value of its neighbors in order to complete its computation. For the edge detection step the size of the neighborhood is dependent on the size of the convolution kernel, e.g. 5×5 for the Nevatia-Babu linear feature extractor [Nevatia and Babu, 1980], selectable for the Canny edge detector [Canny, 1986]. For the edge thinning and edge linking steps the neighborhood is 3×3. Messages are single valued and communication can proceed synchronously among all processes.

For the contour extraction step, no communication among processes is required but, recall that access to the entire 2-D array is required since contours are of "unlimited" extent within the image plane. As we shall see later, when we specify the parallel architecture for the entire process, this situation will change. Partitioning of the low-level algorithms (edge detection, thinning, and linking) will have adverse effects on the parallel implementation of the contour extraction algorithm.

For the linear approximation step, no communication among processes is required. Each contour can be processed independently of all others.

4.5 Architecture Specification

In the architecture specification step our objective is to map each of the processing steps to a portion of the heterogeneous architecture. To perform this mapping, we use the parameters of each of the sections of the architecture such as processing protocol, processing element type, processing element coupling, processor homogeneity, processor synchronicity, and communication network topology. At this point the mappings for each of the individual algorithms that constitute the linear feature extraction process will be independent. Later we will tackle the job of interfacing the various algorithms.

Given the fine granularity, the local neighborhood communication requirements, the synchronous nature of the communications, the simplicity of the required processing, and the data independence of the algorithms, the edge detection, edge thinning, and edge linking steps map well to the SIMD mesh connected section of the architecture. This concurs with the findings of other researchers [Lee and Aggarwal, 1987] [Little *et al.*, 1987]. Contrary to this approach is the one used in [Vaillant *et al.*, 1989] where the algorithms are mapped onto a MIMD based architecture. The authors found that the edge linking process presented a processing bottle neck.

For the contour extraction and linear approximation steps, the parallelism is coarse grain, the algorithms are data dependent, comprise relatively simple operations, and do not require inter-process communication and therefore, map well to the MIMD section. As there is no required interprocess communication, any topology will suffice. The specification of a ring topology will be justified later. This concurs with the implementation described in [Vaillant *et al.*, 1989] in which the authors achieved good performance with an intuitive implementation. Contrary to this approach is that of [Shu *et al.*, 1990] where the authors map these algorithms onto the SIMD section of the IUA. Good performance is achieved but only at the cost of custom hardware and complex software.

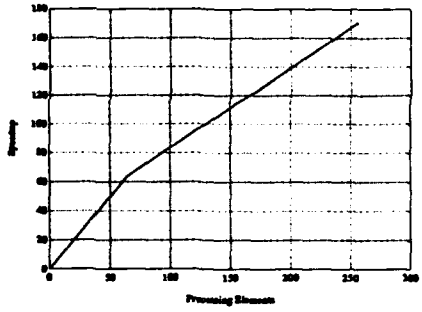
5 Process Interfacing

In utilizing a heterogeneous approach to the parallel implementation of a process one must be cognizant of the interface between the two distinct architectures. For the linear feature extraction process it is at this interface where data is transitioned from the iconic representation, the 2-D array of pixel values, to the symbolic representation, the linked-lists of contours. The contour extraction algorithm performs this transition.

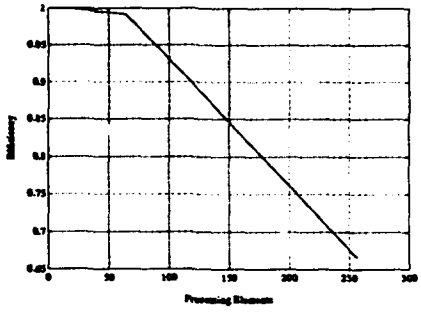
The number of contours detected in a scene will be significantly less than the number of edges detected. This implies that the number of processing elements that can be effectively utilized by the contour extraction and linear approximation algorithms will be significantly less than that used by the edge detection, edge thinning, and edge linking algorithms. Therefore, a pyramid structure between the two architectures is appropriate. This is also the scheme used in the Image Understanding Architecture to interface the CAAPP (SIMD) and ICAP (MIMD) layers. Each MIMD processing element is directly connected to a group of SIMD PEs, and the groups are non-overlapping.

But, with this arrangement a single contour may be distributed across many groups of SIMD PEs, or equivalently, across many MIMD PEs. Therefore, after each MIMD processing element performs the contour extraction algorithm on its partition (data residing in its associated SIMD PEs), and before each performs the linear approximation algorithm on its set of contours, an additional step, *contour merging*, must be performed. Since the distribution of *partial contours* is arbitrary, the MIMD processing elements must perform a *complete exchange* operation in order to gather all of their partial contours.

This is the justification for the ring topology of the MIMD section of the architecture. The complete exchange operation is performed efficiently on a ring topology by "circulating" data values through the ring systolically. Each processing element is made responsible for a set of contours and as sections of those contours are received, they are saved in local memory. Sections of contours that do not belong to the set are passed on. Upon receipt of all sections of its set of contours, each processing element must reconstruct the complete contours by performing the contour merging step. Upon completion, linear approximation can proceed independently for each contour. The contour merging step requires $O(N)$ communication steps for a system consisting of N processing elements. The additional execution time complexity introduced by the contour merging step is discussed in the following section.



(a) Algorithm speedup.



(b) Processor efficiency.

Figure 4: Estimated performance curves for the contour extraction, contour merging, and linear approximation algorithm suite.

6 Performance Analysis

6.1 Complexity Analysis

As shown previously (in tabular form) the time complexity of the entire linear feature extraction process is

$$LFE_{seq} = O(i^2) + O(i^2) + O(i^2) + O(i^2 l) + O(nl)$$

where the terms correspond to the edge detection, edge thinning, edge linking, contour extraction, and linear approximation algorithm complexities respectively.

For synchronous, data independent (SIMD) operations such as those performed by the edge detection, edge thinning, and edge linking algorithms, researchers have shown that high degrees of algorithm speedup and processor efficiency are readily achieved [Little *et al.*, 1987] [Rosenfeld, 1987] [Prasanna-Kumar and Reisis, 1988]. The determining factor for speedup and efficiency is the "degree of match" between the pattern of communication among processes and the interconnect topology of the architecture (solution of the mapping problem.) Our use of the 2-D mesh for the local communication patterns of the linear feature extraction algorithm has been shown by others to perform very well [Lee and Aggarwal, 1987] [Weems *et al.*, 1991]. High degrees of algorithm speedup and processor efficiency are readily achieved.

The operations performed by the contour extraction and linear approximation algorithms do not suffer any overhead due to process communication and, therefore, will achieve high degrees of algorithm speedup and processor efficiency so long as the number of processors is kept in a range bounded by the problem size. Thus, our estimate for the time complexity of the parallel implementation of the five constituent algorithms is

$$LFE_{par} \simeq O(1) + O(1) + O(1)$$
$$+O(i^2 l/N) + O(nl/N)$$

(near linear speedup) given an $i \times i$ SIMD mesh and N MIMD processing elements.

The open issue regarding the performance of the parallel implementation of the linear feature extraction algorithm is the time complexity introduced by the inclusion of the contour merging step and its associated communication requirements. This is where we focus our attention.

To form complete contours from its set of partial contours, each processing element must perform the following algorithm:

```
(contour merging)
for each sub-contour, i
  change ← TRUE
  while change
    change ← FALSE
    for each sub-contour, j ≠ i
      if head(i) = tail(j)
        connect(i, j)
        change ← TRUE
      else if tail(i) = head(j)
        connect(j, i)
        change ← TRUE
```

The algorithm traverses the list of partial contours (sub-contours) combining spatially adjacent sub-contours reducing the length of the list by one every iteration. Given that a contour is partitioned into n sub-contours, the time complexity of the contour merging algorithm is

$$MERGING = O(\sum_{i=1}^{n} i) = O(\frac{n(n+1)}{2}).$$

Then the estimate made above (near linear speedup) for the time complexity of the entire parallel implementation of the linear feature extraction process is amended with this new term to

$$LFE_{par} = O(1) + O(1) + O(1)$$
$$+O(i^2 l/N) + O(\frac{n(n+1)}{2}) + O(nl/N).$$

The open issue is thus reduced to one of comparing

$$O(i^2 l/N) + O(\frac{n(n+1)}{2}) + O(nl/N)$$

to

$$O(i^2 l) + O(nl)$$

for various numbers of processing elements, N, and contours, n. Although the length and number of subcontours that a single contour is partitioned into are also critical, they are bounded by the image size, i, and the number of MIMD processing elements. Figure 4 shows the comparison of these terms along with the estimated algorithm speedup and processor efficiency for the parallel implementation. We have assumed that all contours are the same length and that each contour is "maximally" fragmented (partitioned into N, the number of MIMD processing elements) which is the worst case. The shape of these curves indicates that for a 256×256 processing element SIMD mesh, 64 MIMD PEs can be effectively utilized before performance begins to degrade. We use these curves for comparison to simulation data in the next section.

6.2 Observed Performance

Figure 5 shows the partitioning of an aerial image into areas covered by the individual 64 MIMD processing elements. From this figure one can imagine how increasing the number of MIMD processing elements would fragment long contours thus affecting the performance of the contour merging step. Also, by decreasing the number of MIMD PEs each would be overworked.

Figure 6 shows the observed performance curves (from simulation of the parallel implementation) for the aerial image. The simulation reflects time required by the algorithm as well as overhead due to communication and data dependencies. Since this overhead is included in the simulation the shape of the curves differ from that of the estimated curves but the same trends exist. That is, performance degrades when the number of MIMD processing elements is increased beyond 64 processing elements.

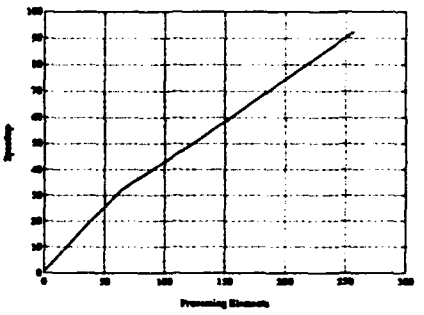
6.3 Algorithm Structure

Recall that one of our goals is to design a parallel implementation that agrees with one's intuition. That is, one that is not complicated by the underlying architecture. We feel that this is important because of the modularity of the linear feature extraction process and the possibility of changing constituent algorithms at a later date.

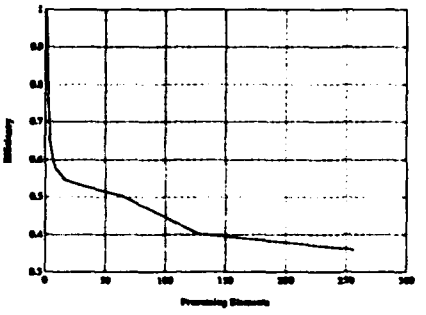
With regard to the structure of the algorithm and our resultant implementation, our primary concern for the linear feature extraction process is the addition of the contour merging step. This addition is somewhat contrary to our desire of retaining the basic structure of the algorithm. But, a trade-off is made that is beneficial to the run-time performance and not too detrimental to life cycle issues. The addition of the contour merging step is such that it allows the original algorithms (edge detection, edge thinning, edge linking, contour extraction, and linear approximation) to achieve significant algorithm speedup and processor efficiency through intuitive and modular implementations. The additional step is strictly a "mutually exclusive" addition in that it does not affect the implementations of the originals steps. The structure of the parallel implementation of the entire process still resembles the structure of an intuitive implementation with the addition of the contour merging step. Thus, the effort (cost) required to realize and maintain the parallel implementation is only slightly greater than that of the sequential implementation.



Figure 5: Image plane partitioning for 64 MIMD processing elements.



(a) Algorithm speedup.



(b) Processor efficiency.

Figure 6: Performance curves from simulation of the parallel linear feature extraction implementation on the aerial image.

7 Summary

In this paper we have described a parallel implementation of the linear feature extraction process. The problem has not been extensively studied in that most work stops short at parallel line "finders", utilizing techniques such as the Hough Transform [Weems *et al.*, 1991] [Guerra and Hambrusch, 1989], or simply edge detectors [Little *et al.*, 1987] [Lee and Aggarwal, 1987] [Weems *et al.*, 1991]. In these studies the heterogeneous, fine/coarse grain nature of the problem is not high-lighted. In studies that have considered the entire problem, they utilized a single parallel processing architecture that is suited to either the fine or the coarse grain algorithms and designed mapping schemes of both types of algorithms onto that one architecture. In doing so, they either had to develop hardware specific implementations [Shu *et al.*, 1990] or settle for reduced performance in some of the constituent algorithms [Vaillant *et al.*, 1989].

In our implementation we have attempted to combine the "best of both worlds" through the use of a heterogeneous architecture. We achieve good algorithm speedup and processor efficiency for all of the constituent algorithms and the process as a whole. Furthermore, the only deviation from the process specification is the addition of an algorithm to "bridge" the fine and coarse grain processes (architectures.) The architecture we utilize is a heterogeneous pyramid with a 64-to-1 reduction from the fine grain (SIMD) to the coarse grain (MIMD) sections which concurs with the findings of the Image Understanding Architecture (IUA) designers in bridging low and mid-level parallel vision processing. The difference between our design and one for the IUA described in [Shu *et al.*, 1990] is our independence on custom "functionality" within the SIMD level. We were able to achieve this by mapping the constituent algorithms onto appropriate parallel processor architectures.

References

[Canny, 1983] J.F. Canny. Finding edges and lines in images. Technical report, Massachusetts Institute of Technology, June 1983. Artificial Intelligence Laboratory.

[Canny, 1986] J.F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, November 1986.

[Guerra and Hambrusch, 1989] C. Guerra and S. Hambrusch. Parallel algorithms for line detection on a mesh. *Journal of Parallel and Distributed Computing*, 6(2):1–19, 1989.

[Lee and Aggarwal, 1987] S.Y. Lee and J.K. Aggarwal. Parallel 2-D convolution on a mesh connected array processor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(9):590–594, July 1987.

[Little *et al.*, 1987] J. J. Little, G. Blelloch, and T. Cass. Parallel algorithms for computer vision on the Connection Machine. In *Proceedings of the DARPA Image Understanding Workshop*, pages 628–638, February 1987.

[Marr and Hildreth, 1980] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London, B*, 207:187–217, 1980.

[Nevatia and Babu, 1980] R. Nevatia and K. R. Babu. Linear feature extraction and description. *Computer Vision, Graphics, and Image Processing*, 13:257–269, 1980.

[Prasanna-Kumar and Reisis, 1988] V.K. Prasanna-Kumar and D. Reisis. Parallel architectures for image processing and vision. In *Proceedings of the DARPA Image Understanding Workshop*, pages 609–619, April 1988.

[Reinhart and Nevatia, 1990] C. Reinhart and R. Nevatia. Efficient parallel processing in high level vision. In *Proceedings of the DARPA Image Understanding Workshop*, pages 829–839, September 1990.

[Rosenfeld, 1987] A. Rosenfeld. A report on the DARPA image understanding architectures workshop. In *Proceedings of the DARPA Image Understanding Workshop*, pages 298–302, February 1987.

[Shu *et al.*, 1990] D.B. Shu, J.G. Nash, M.M. Eshaghian, and K. Kim. Straight-line detection on a gated-connection VLSI network. In *Proceedings of the Tenth Internation Conference on Pattern Recognition*, pages 456–461, June 1990.

[Vaillant *et al.*, 1989] R. Vaillant, R. Deriche, and O. Faugeras. 3D vision on the parallel machine CAPITAN. In *International Workshop on Industrial Applications of Machine Intelligence and Vision*, pages 326–331, April 1989.

[Weems and Levitan, 1987] C. C. Weems and S. P. Levitan. The Image Understanding Architecture. In *Proceedings of the DARPA Image Understanding Workshop*, pages 483–496, February 1987.

[Weems *et al.*, 1991] C. Weems, E. Riseman, and A. Hanson. The DARPA image understanding benchmark for parallel computers. *Journal of Parallel and Distributed Computing*, 11(1):1–24, 1991.

Parallel Algorithms for Stereo and Image Matching

Ashfaq Khokhar and Viktor K. Prasanna *
Department of EE-Systems, EEB 244
University of Southern California
Los Angeles, CA 90089-2562
email: {ashfaq + prasanna}@halcyon.usc.edu

Abstract

In this paper, we summarize our progress in parallelizing two high level vision tasks; stereo matching and image matching. We show processor-time optimal algorithms for stereo and image matching using linear features as primitives using the matching techniques developed by the vision group at USC. These algorithms are designed to execute on fixed size arrays.

1 Introduction

Parallel processing has been used in computer vision over the past two decades. However, most of these solutions have addressed problems in low-level and mid-level vision [12]. This paper presents a summary of our research in parallelizing two high-level vision tasks; stereo and image matching using linear features as primitives. Stereo matching is one of the well known methods for extraction of depth information. Depth recovery is a crucial problem in image understanding with applications in robotics and navigation. Also, image matching is a fundamental operation in machine vision and is a key step in object recognition.

For stereo matching, we propose $O(\frac{Nn^3}{P})$ time algorithm on a P processor linear array, where N is the number of line segments in one image, n is the number of line segments in a window determined by the object size, and $P \leq n$. This algorithm is extended to a mesh array of $P \times P$ processors to run in $O(\frac{Nn^3}{P^2})$ time. The sequential algorithm takes $O(Nn^3)$ time. For image matching, we first propose a fast sequential algorithm which runs in $O(n^2m^2)$ time, where n is the number of line segments in the image and m is the number of line segments in the model. Previously known approaches to the image matching problem take $O(n^3m^3)$ time. A parallel algorithm is developed using the proposed sequential algorithm. $O((\frac{nm}{P} + P)nm)$ time performance is achieved on a P processor fixed size linear array, where $P \leq nm$. This leads to a processor-time optimal solution for $P \leq \sqrt{nm}$. Also, a mesh algorithm for image matching is derived which runs in $O((\frac{nm}{P} + P)nm)$ time performance is achieved on a $P \times P$ processor array. All the proposed parallel algorithms achieve linear speed-up compared with the corresponding sequential algorithms.

*This research was supported by the Defense Advanced Research Projects Agency under contract F49620-90-C-0078, monitored by the Air Force Office of Scientific Research. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

2 Fixed Size Arrays

Two parallel architectures are used in our algorithms:

A: *Fixed Size Mesh Array*: A *fixed size mesh array* is a two dimensional array of $P \times P$ processors, where P^2 is less than or equal to the problem size. Each processor PE_{ij} is connected to PE_{i+1j}, PE_{i-1j}, PE_{ij-1}, and PE_{ij+1}, if they exist. A memory plane of $P \times P$ memory modules (MMs) is provided. Each PE_{ij} is attached to memory module MM_{ij}. The architecture is shown in Fig. 1.

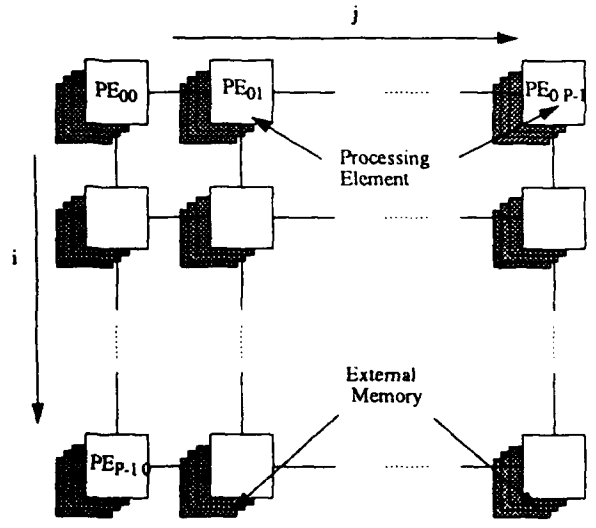


Figure 1: A Fixed Size Mesh Array

B: *Fixed Size Linear Array*: A *fixed size linear array* is a one dimensional array of processors, say P, where P is less than or equal to the problem size. Processor i is denoted as PE_i, where $0 \leq i \leq P-1$. Each PE_i is attached to an external memory module (MM_i). The architecture is shown in Fig. 2.

In both the models, processors are connected through bidirectional local links and the arrays operate in SIMD

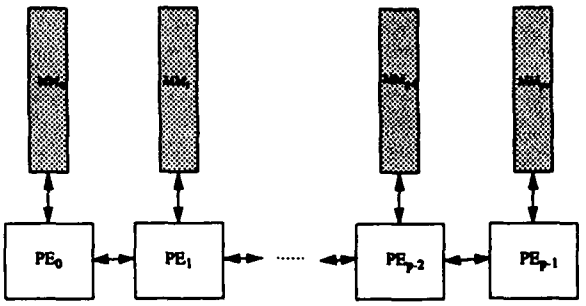


Figure 2: A Fixed Size Linear Array.

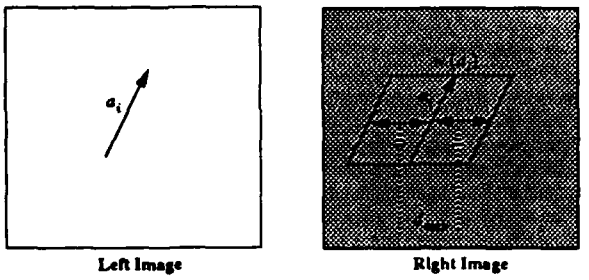


Figure 3: Determining a *Stereo-window*

mode. Following assumptions are made regarding the computations in these models:

- Each arithmetic/logic operation performed in a PE takes $O(1)$ time.
- Each access by PE_i to memory module MM_i, $0 \leq i \leq P-1$, takes $O(1)$ time.
- Each access by PE_{ij} to memory module MM_{ij}, $0 \leq i,j \leq P-1$, takes $O(1)$ time.
- A unit data transfer between adjacent PEs takes $O(1)$ time.
- Each PE has indirect addresssing capability.

3 Stereo Matching on Fixed Size Arrays

In stereo matching, two images, left and right image, captured at the same time but at different angles are matched. Various stereo matching algorithms differ with respect to the primitives used for matching [3]. Each technique has its own advantages and disadvantages. Stereo matching using linear features is capable of handling more complex scenes (such as those containing repetitive structures) [9]. In this section we provide a fast parallel implementation of the stereo matching algorithm (also called the Minimum Disparity Algorithm) described in [9]. For the sake of completeness, the main ideas of this algorithm are presented in Section 3.1.

3.1 Minimum Disparity Algorithm

The technique attempts to match overlapping segments detected along the same epipolar line, having similar contrast and orientation. Following the terminology in [9], for each segment a_i in the left image, a match is found in a window $w(a_i)$ defined in the right image. Similarly, for each segment b_j in the right image, a match is found in $w(b_j)$ defined in the left image. The shape of the window is a parallelogram. This is shown in Fig. 3 for left to right match, one side corresponds to a_i, and the other side is a horizontal vector of length $2d_{max}$, where d_{max} is the maximum disparity. The number of segments in each window is assumed to be at most n and both the images are assumed to have N segments each.

For each a_i, a set $S_p(a_i)$ of possible matches in window $w(a_i)$ is defined based on the contrast, overlap, and orientation. Similarly for each b_j, a set $S_p(b_j)$ is defined. To assign unambiguous matches, a set of matches is considered together for each segment in the image. For each possible element j in $S_p(a_i)$, an evaluation function $v(i,j)$, is computed, which is dependent on how well the disparities of the other line segment matches in $w(b_j)$ agree with the average disparity d_{ij} of the matching pair. A set of preferred matches $Q^t(a_i)$ is constructed for each i during iteration t, if the following holds:

$$\forall k \in S_p(a_i) \text{ such that } b_k \leftrightarrow b_j, v^t(i,j) < v^t(i,k) \quad (1)$$

and

$$\forall h \in S_p(b_j) \text{ such that } a_h \leftrightarrow a_i, v^t(i,j) < v^t(h,j)\ . \quad (2)$$

The relation $b_k \leftrightarrow b_j$ is true if b_k overlaps b_j.

$v(i,j)$ is defined as follows:

$$v^{t+1}(i,j) = \sum_{a_h \text{ in } w(b_j)} \min_{b_k \text{ verifies } C_1(a_h)} \lambda_{ijhk} \left| d_{hk} - d_{ij} \right| / card(b_j)$$

$$+ \sum_{b_k \text{ in } w(a_i)} \min_{a_h \text{ verifies } C_2(b_k)} \lambda_{ijhk} \left| d_{hk} - d_{ij} \right| / card(a_i)$$

In the above equation, $t+1$ indicates the iteration number and $\lambda_{ijhk} = \min(\text{overlap}(i,j), \text{overlap}(h,k))$ and $card(a_i)$ is the number of segments in $w(a_i)$. The relations C_1 and C_2 are defined as follows. We say b_k verifies $C_1(a_h)$ if:

1. If $Q^t(a_h) \neq \emptyset$, b_k is in $Q^t(a_h)$ else b_k is in $S_p(a_h)$,
2. Either $b_k \neq b_j$, or a_h and a_i do not overlap.

The algorithm terminates after a constant number of iterations [9] and each iteration takes $O(Nn^3)$ time. Each time unit corresponds to a simple arithmetic/logic operation. A formal description of the algorithm is given in Fig. 4. Also, procedure *i-pref(i)* mentioned in Fig. 4 in shown in Fig 5. Similar procedure for *j-pref(j)* can be developed.

3.2 Parallel Stereo Matching

In this section, we first give a parallel version of the algorithm described in Fig. 4 and then provide parallel implementations of procedure *i-pref(i)* on fixed size arrays. We have also devised an efficient data partitioning strategy for the stereo matching problem.

In Fig. 6, procedures *Parallel-i-pref(i)* and *Parallel-j-pref(j)* determine the partial *preferred* matches, $QT(a_i)$

```
1. repeat
2.     change ← 0;
3.     for i = 1 to N do
4.         for each j such that j ∈ w(a_i) do
5.             i-pref(i, j, QT(a_i))
6.         end
7.     end;
8.     for j = 1 to N do
9.         for each i such that i ∈ w(b_j) do
10.            j-pref(j, i, QT(b_j))
11.        end
12.    end;
13.    for i = 1 to N do
14.        for each j such that j ∈ w(a_i) do
15.            Q-update(i, j)
16.        end
17.    end
18.    for i = 1 to N do Q(a_i) ← Q'(a_i);
19.    for j = 1 to N do Q(b_j) ← Q'(b_j);
20. until (change = 0)
```

Figure 4: Sequential Algorithm for Stereo Matching

and $QT(b_j)$, for a_i and b_j respectively. The third procedure *Parallel-Q-update* is used to determine the new sets of preferred matches, $Q(a_i)$ and $Q(b_j)$, for a_i and b_j, respectively. In the following sections, $\frac{a}{b}$ denotes a mod b.

3.2.1 Data Partitioning

In stereo matching, input to the algorithm is a set of N segments from the right image and a set of N segments from the left image. As described in Section 3.1, each segment is represented by its length, contrast and orientation. With each pair (a_i, b_j), such that a_i, b_j overlap and have similar contrast and similar orientation, an average disparity d_{ij} is associated. In order to find a possible match for each segment a_i in the left image, a window $w(a_i)$ is defined in the right image. Similarly, for each segment b_j in the right image a window $w(b_j)$ is defined in the left image. Each window is assumed to contain at most n segments. An efficient partitioning algorithm is devised in [5]. The running time of the algorithm is $O(Nn \log P)$.

3.2.2 Partitioned Implementation on a Fixed Size Mesh Array

Based on the algorithm shown in Fig. 5, parallel algorithm on a $P \times P$ mesh array is developed. Similar procedure for *Parallel-j-pref*(j) can be developed.

In each i-loop, $\mathrm{PE}_{\frac{h}{p}\frac{j}{p}}$, $0 \le h, j \le n-1$ (or $\mathrm{PE}_{\frac{h}{p}\frac{j}{p}}$ for the second part of the j-loop) is used for the computation of $\min(i, j, k)$ (or $\min(i, j, h)$). The information required to accomplish the computation in $\mathrm{PE}_{\frac{h}{p}\frac{j}{p}}$ (or $\mathrm{PE}_{\frac{h}{p}\frac{j}{p}}$) includes:

1. if $t = 0$, then $S_p(a_h)$ (or $S_p(b_k)$), else $Q^t(a_h)$ (or $Q^t(b_k)$),
2. $d_{hk}, 0 \le k \le n-1$ (or $d_{hk}, 0 \le h \le n-1$) and
3. $\lambda_{ijhk}, 0 \le k \le n-1$ (or $\lambda_{ijhk}, 0 \le h \le n-1$)

```
procedure i-pref(i, j, QT(a_i))
1. for each h such that h ∈ w(b_j) do
2.     for each k such that b_k verifies C1(a_h) do
3.         min_h ← min(min_h, λ_ijhk|d_hk − d_ij|);
4.     end;
5.     sum1(i, j) ← sum(i, j) + min_h;
6. end;
7. ave1(i, j) ← sum1(i, j)/card(b_j);
8. for each k such that k ∈ w(a_i) do
9.     for each h such that a_h verifies C2(b_k) do
10.        min_k ← min(min_k, λ_ijhk|d_hk − d_ij|);
11.    end;
12.    sum2(i, j) ← sum(i, j) + min_k;
13. end;
14. ave2(i, j) ← sum2(i, j)/card(a_i);
15. sum(i, j) ← ave1(i, j) + ave2(i, j);
16. case:
17.    sum(i, j) < Min(i):
18.        QT(a_i) ← { j };
19.        Min(i) ← sum(i, j);
20.    sum(i, j) = Min(i):
21.        QT(a_i) ← QT(a_i) ⋃ { j };
22. end
```

Figure 5: Finding Partially Preferred Matches for the Left Image

```
1. repeat
2.     change ← 0;
3.     for i = 1 to N do
4.         Parallel-i-pref(i);
5.     for j = 1 to N do
6.         Parallel-j-pref(j);
7.     for i = 1 to N do
8.         Parallel-Q-update(i);
9. until (change = 0)
```

Figure 6: Parallel Algorithm for Stereo Matching

The main steps of procedure *Parallel-i-pref*(i) are briefly discussed in the following, with the corresponding execution time indicated within parentheses.

1. $\forall\ j \in w(a_i)$, load d_{hk} and λ_{ijhk}, $O(n)$ data to $\mathrm{PE}_{\frac{h}{p}\frac{j}{p}}$, $O(n^3)$ data in all. $(O(\frac{n^3}{P^2}))$
2. PE_{00} broadcasts d_{ij} to all the processors in the array. $(O(P))$
3. Perform the *min* operation over all k to determine $\min(i, j, h)$ in $\mathrm{PE}_{\frac{h}{p}\frac{j}{p}}$. $(O((\frac{n}{P})n))$
4. Along each column of processors, i.e. $\forall j$, all the $\min(i, j, h)$ values are summed up and saved in $\mathrm{PE}_{P-1\frac{j}{p}}$. $(O((\frac{n}{P})P))$
5. In each $\mathrm{PE}_{P-1\frac{j}{p}}$, compute the average.
6. $\forall\ j \in w(a_i)$, load d_{hk} and λ_{ijhk}, $O(n)$ data to

$PE_{\frac{k}{p}\frac{j}{p}}$, $O(n^3)$ data in total. $(O(\frac{n^3}{P^2}))$

7. Perform the *min* operation over all h to determine $\min(i,j,k)$ in $PE_{\frac{k}{p}\frac{j}{p}}$. $(O((\frac{n}{p})n))$
8. Along each column of processors, i.e. $\forall j$, all the $\min(i,j,k)$ values are summed up and saved in the last processor $PE_{P-1\frac{j}{p}}$. $(O((\frac{n}{p})P))$
9. In each $PE_{P-1\frac{j}{p}}$, $\forall\ j \in w(a_i)$, compute the average of the sum obtained in step 8 and add to the average obtained in step 5. $(O(1))$
10. Along the last row of processors, find the minimum of all the values obtained in step 9, and store the corresponding b_j back in the memory, which is $QT(a_i)$. $(O(n))$

Similar steps can be designed for the procedure *Parallel-j-pref(j)*.

It can be easily verified that for each a_i the procedure *Parallel-i-pref(i)* runs in $O(n^3/P^2)$ time with each time unit corresponding to a simple arithmetic/logic operation. The resulting $QT(a_i)$'s and $QT(b_j)$'s can then be combined in constant time by using the procedure *Parallel-Q-update*. Therefore, each iteration takes $O(\frac{Nn^3}{P^2})$ time.

3.2.3 Partitioned Implementation on a Fixed Size Linear Array

The main steps of procedure *Parallel-i-pref(i)* on a fixed size linear array of P processors are briefly discussed in the following, with the corresponding execution time indicated within the parentheses.

1. $\forall\ h \in w(b_j)$, load d_{hj} and λ_{ijhk}, $O(n)$ data to $PE_{\frac{h}{p}}$, $O(n^2)$ data in all. $(O(\frac{n^2}{P}))$
2. PE_0 broadcasts d_{ij} to all the PEs, $0 \le i,j \le n-1$. $(O(P))$
3. Perform the *min* operation over all k to determine $\min(i,j,h)$ in $PE_{\frac{h}{p}}$. $(O((\frac{n}{p})n))$
4. Along the linear array, i.e. $\forall h$, all $\min_h$ values are summed up and saved in PE_0. $(O((\frac{n}{p})P))$
5. Compute the average in PE_0.
6. $\forall\ k \in w(a_i)$, load d_{kj} and λ_{ijhk}, $O(n)$ data to $PE_{\frac{k}{p}}$, $O(n^2)$ data in all. $(O(\frac{n^2}{P}))$
7. $\forall\ k \in w(a_i)$, perform the *min* operation over all h to determine $\min(i,j,k)$ in $PE_{\frac{k}{p}}$. $(O((\frac{n}{p})n))$
8. Along the linear array, i.e. $\forall k$, all $\min_k$ values are summed up and saved in the PE_0. $(O((\frac{n}{p})P))$
9. Take the average of the sum obtained in step 8 and add it to the average obtained in step 5.
10. $\forall j$, find the minimum of all the values obtained in step 9, which is $QT(a_i)$ and store it back in the memory. $(O(n))$

Similar steps can be designed for procedure *linear-j-pref(j)*.

For each i, PE_1 is assigned to store the value $QT(a_i)$ obtained at the end of the procedure. It can be easily verified that for each i the procedure *Parallel-i-pref(i)* runs in $O(n^3/P)$ time.

All the resulting $QT(a_i)$'s and $QT(b_j)$'s can then be combined in constant time by using the procedure *Parallel-Q-update*. Therefore, each iteration takes $O(Nn^3/P)$, which represents optimal speedup.

For details of these implementations, refer to [5; 6].

4 Image Matching on Fixed Size Arrays

Image matching problem plays a key role in object recognition. In the past, several approaches have been proposed for this problem [13; 15; 2], which, in general, differ with respect to the primitives used for matching. In this section, we consider image matching using linear features [10] for parallel implementation. Readers can refer to [10] for additional details of the matching technique. We begin with the basic idea of this approach and then present processor-time optimal parallel algorithms on fixed size arrays.

4.1 Matching Technique

In general, in the image matching problem, we have n objects, $\{o_1, o_2, \ldots, o_n\}$, in the scene and m labels, $\{l_1, l_2, \ldots, l_m\}$, in the model. Here, the objects are *segments* in the scene derived from edge detectors and are described by the coordinates of their end points, orientation and average contrast. The matching technique computes the quantity v_{ip}, in $\{0,1\}$, which is the possibility of assigning label l_p to object o_i.

The method [10] relies on geometric constraints, which means that when a label l_p is assigned to object o_i, we expect to find an object o_j with assigned label l_q in an area depending on i,p,q. The *match-window* $W(i,p,q)$ denotes the area described by the parameters i,p,q. By representing the object o_i with a vector $\vec{A_iB_i}$, the label l_p with $\vec{C_pD_p}$ and label l_q with $\vec{C_qD_q}$, we can determine the four extreme points, W_1, W_2, W_3, W_4, of the induced *match-window* $W(i,p,q)$ using the following relations: (μ denotes the scaling factor known beforehand)

- $\vec{A_iW_1} = \mu \cdot \vec{C_pC_q}$, $\vec{W_1W_2} = \mu \cdot \vec{C_qD_q}$
- $\vec{B_iW_3} = \mu \cdot \vec{D_pC_q}$, $\vec{W_3W_4} = \mu \cdot \vec{C_qD_q}$

Fig. 7 shows the relationship between the window and the segments.

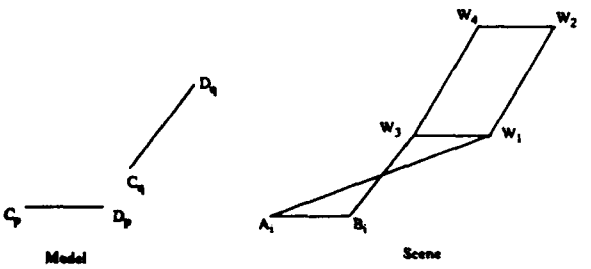


Figure 7: Determining a *match-window*

The meaning of *compatibility* is defined as follows:

$< i,p >$ is *compatible* with $< j,q >$ *iff* o_i in $W(j,q,p)$ and o_j in $W(i,p,q)$,

Let $\Omega^x_{ij}[p,q]$ denote the *compatibility* of assigning label l_p to object i and label l_q to object j. A weak notion of consistency is used to determine whether an assignment is *feasible*. A predetermined confidence factor, $\delta \leq m$, is used to decide the *feasibility* of $< i,p >$ as in the following update statement during an iteration:[1]

> For every i,p, $v'_{ip} \leftarrow v_{ip}$ **AND** 'condition A', where 'condition A' is true if ($\exists S \subseteq \{1,2,\ldots,m\}$ and $\|S\| = \delta$, such that for every $q \in S$, $\exists j \in \{1,2,\ldots,n\}$ such that $v_{jq} = 1$ and $\Omega^x_{ij}[p,q] = 1$) and is false otherwise.

The algorithm stops when for all i,p, $v'_{ip} = v_{ip}$. We can rewrite the above update statement as

$$v'_{ip} \leftarrow v_{ip} * \bigwedge_{q=1}^{\delta,m} [\sum_{j=1}^{n} (v_{jq} * \Omega^x_{ij}[p,q])], \qquad (3)$$

where

$$\bigwedge_{q=1}^{\delta,m} X_q = \begin{cases} 1 & \text{if } \sum_{q=1}^{m} X_q \geq \delta \\ 0 & \text{otherwise .} \end{cases} \qquad (4)$$

Note that the operation $\sum_{j=1}^{n}$ in equation 3 is a logical OR operation, while the operation $\sum_{q=1}^{m} X_q$ in equation 4 is an arithmetic ADD operation.

With the modified update statement given by equation 4, we have designed a faster sequential algorithm which is easier to parallelize compared with the one proposed in [10]. This algorithm is an extension of the discrete relaxation algorithm developed in [8] and it takes $O(n^2m^2)$ time. Each time unit corresponds to a simple arithmetic/logic operation. The original algorithm [10] runs in $O(n^2m^2dw)$ time, where d is the density of the segments and w is the window size. In the worst case, d and w can be n and m respectively. For more details refer to [5].

4.2 Parallel Image Matching

In this section, we present parallel algorithms for the image matching problem. Since the size of a *match-window* determined by any object and two labels is much smaller than the size of the complete image, the number of initially assignable segments in any of the *match-windows* for each object is much smaller than the total number of objects in the image. This allows us to obtain a partitioned implementation in which each processor is responsible for more than one <object, label> pair.

A parallel algorithm is shown in Fig. 8. Each v_{ip} is associated with $m+1$ counter variables. These are $N_{ip}[q], 0 \leq q \leq m-1$, and T_{ip}. These counter variables have the following definitions:

- $N_{ip}[q]$ denotes the number of 1's in the n entries of $\Omega^x_{ij}[p,q]$, $0 \leq j \leq n-1$, and
- T_{ip} denotes the number of nonzero $N_{ip}[q]$'s, $0 \leq q \leq m-1$.

```
    { Initialization }
1.  Initialize all Ω^x_ij[p,q]'s in parallel;
2.  parallel do (in PE_ip, 1 ≤ i ≤ n, 1 ≤ p ≤ m)
3.     v_ip ← 1;
4.     for q = 1 to m do
5.        N_ip[q] ← 0;
6.        T_ip ← 0
7.        for j = 1 to n do
8.           if (Ω^x_ij[p,q] = 1) then N_ip[q] ← N_ip[q] + 1;
9.        end;
10.       if (N_ip[q] ≠ 0) then T_ip ← T_ip + 1;
11.    end;
12.    if (T_ip < δ) then do
13.       Send_ip ← 1;
14.       v_ip ← 0;
15.    end;
13. parallel end ;

    { Iteration }
16. repeat
17.    parallel do (in PE_ip, 1 ≤ i ≤ n, 1 ≤ p ≤ m)
18.       if (Send_ip = 1) then
19.          send Id < i,p > to all the PEs;
20.       { if (no broadcast Id acknowledged)
21.         then stop;
22.         else a broadcast Id, say < j,q >, is
            acknowledged by all PE's;}*
23.       if (< i,p > = < j,q >) then Send_ip ← 0;
24.       if ((v_ip = 1) AND (Ω^x_ij[p,q] = 1)) then do
25.             N_ip[q] ← N_ip[q] − 1;
26.             if (N_ip[q] = 0) then T_ip ← T_ip − 1;
27.             if (T_ip < δ) then do
28.                Send_ip ← 1;
29.                v_ip ← 0;
30.             end
31.          end
32.       end
33.    parallel end
34. forever
    *: the code inside braces is the broadcast operation.
```

Figure 8: Parallel Algorithm for Image Matching

For v_{ip}, each of the m $N_{ip}[q]$'s are used to determine an object for label l_q, i.e., if we can find any object to be labelled with l_q when o_i is labelled with l_p. T_{ip} is used to determine if there are at least δ such *compatible* labelings when o_i is labelled with l_p. Each infeasible pair $< i,p >$ (having $T_{ip} \leq \delta$) is broadcast to all PEs. Each PE checks if it has any pair $< j,q >$ such that $\Omega^x_{ij}[p,q] = 1$ and decrements $N_{ip}[q]$. At any time, if $N_{ip}[q]$ becomes zero, T_{ip} is decremented. As a result, if $T_{ip} \leq \delta$, pair $< j,q >$ is marked *infeasible*.

4.2.1 Partitioned Algorithms on Fixed Size Arrays

Based on the algorithm shown in Fig 8, partitioned algorithms on fixed size mesh and linear arrays are obtained. These algorithms differ with respect to data routing schemes.

In a mesh array of $P \times P$ processors, each of the P^2 PEs process $\frac{nm}{P^2}$ distinct v_{ip} values ($\frac{nm}{P}$ values in case of

[1] $\|S\|$ denotes the cardinality of S

linear array). The data stored in each memory module include $\frac{nm}{P^2}$ v_{ip} values ($\frac{nm}{P}$ in linear array), corresponding nm $\Omega^x_{ij}[p,q]$ values, m $N_{ip}[q]$ counter values and the T_{ip} variable. Also, a flag is stored in each MM for each v_{ip} to indicate whether the *infeasibility* has been *acknowledged* by all the PEs. Such an acknowledgment triggers the necessary update of the corresponding counters in each PE. Also, in each PE, an extra flag is used to indicate if at least one such *infeasible* assignment is yet to be *acknowledged*.

An initialization procedure is executed in each PE to initialize the m counter variables for each of its v_{ip} values. Based on the condition defined in Section 4.1, each PE sets the corresponding flag for an *infeasible* assignment and retains the Id of one such assignment for later broadcast. This can be performed in $O(\frac{nm}{P^2})$ time on a $P \times P$ mesh ($O(\frac{nm}{P})$ time on the linear array). During each iteration, a 'collect' operation is first executed. The purpose of this operation is to gather the Ids retained in all the PEs at the end of the previous iteration. The 'collect' operation can be executed in $O(P)$ time on both arrays. Details of the data routing algorithms can be found in [5; 6].

The execution time of the mesh algorithm is $O((\frac{nm}{P^2} + P)nm)$. This implementation leads to a processor-time optimal solution when $P \leq (nm)^{1/3}$. Additional details can be found in [6].

The total execution time of the linear array algorithm is $O((\frac{nm}{P} + P)nm)$. This implementation leads to a processor-time optimal solution when $P \leq \sqrt{nm}$. Further details can be found in [5].

5 Conclusions

Image matching and stereo matching are key problems in image understanding. In this paper we have summerized our work [5; 6] in parallelizing a well known matching technique developed by the vision group at USC. Several other sequential approaches to these problems have been proposed. These solutions vary mainly with respect to the primitives used for matching. Additional work is needed to consolidate these approaches and provide a uniform framework for parallel stereo and image matching.

References

[1] H. Baker and T. Binford, *"A System for Automated Stereo Mapping"*, Proc. Image Understanding Workshop, Palo Alto, CA, 1982.

[2] C. Clark, A. Luk, and C. McNary, *"Feature based Scene Analysis and Model Matching"*, Proc. of NATO Advanced Study Inst. for Pattern Recognition and Signal Processing, France, 1978.

[3] U. Dhond and J. K. Aggarwal, *"Structure from Stereo - A Review"*, IEEE Trans. on Systems, Man and Cybernetics, Nov. 1989.

[4] M. Drumheller and T. Poggio, *"On Parallel Stereo"*, Proc. IEEE Conference on Robotics and Automation, 1986.

[5] A. Khokhar, W. Lin, and V. K. Prasanna, *"Stereo and Image Matching on Fixed Size Linear Arrays"*, submitted to CVPR '92.

[6] A. Khokhar, W. Lin, and V. K. Prasanna, *"Stereo and Image Matching on Fixed Size Mesh Based Arrays"*, In the Proc. of Computer Architectures for Machine Perception, 1991 (CAMP '91).

[7] A. Khokhar, *"Parallelism in Stereo and Image Matching"*, Ph.D. Thesis, in preparation, Dept. of EE-Systems, University of Southern California, Los Angeles.

[8] W. Lin and V. K. Prasanna Kumar, *"Parallel Algorithms and Architectures for Discrete Relaxation Technique"*, Proc. of IEEE Conference on Vision and Pattern Recognition, 1991.

[9] G. Medioni and R. Nevatia, *"Segment-Based Stereo Matching"*, Computer Vision, Graphics, and Image Processing, Feb. 1985.

[10] G. Medioni and R. Nevatia, *"Matching Images Using Linear Features"*, IEEE Trans. on PAMI, Nov. 1984.

[11] D. Nassimi and S. Sahni, *"Data Broadcasting in SIMD Computers"*, IEEE Trans. on Computers, 1981.

[12] V. K. Prasanna Kumar, Ed., "Parallel Algorithms and Architectures for Image Understanding", Academic Press, Boston, 1991.

[13] K. Price, *"Symbolic Matching of Images and Scene Models"*, Proc. of IEEE Workshop on Computer Vision, Aug. 1982.

[14] C. Reinhart and R. Nevatia, *"A Prescription for the Perils of Parallelism in High Level Computer Vision"*, submitted to IEEE Computer, 1991.

[15] L. Shapiro and R. Haralick, *"Structural Description and Inexact Matching"*, IEEE Trans. on PAMI, Sept. 1981.